UNIVERSIDAD COMPLUTENSE DE MADRID

FACULTAD DE INFORMÁTICA

DPTO DE INGENIERÍA DEL SOFTWARE E INTELIGENCIA ARTIFICIAL

Sistemas Informáticos 2010/2011

Sistema de Visión Estereoscópica para Navegación Autónoma de vehículos no tripulados

Por:

Juan Luis Trincado Alonso
Beatriz Torres Salcedo
Ana Pérez Alonso

Profesor director: Gonzalo Pajares Martinsanz

ÍNDICE

Declaración de conformidad

Los alumnos:

Juan Luis Trincado Alonso, Beatriz Torres Salcedo, Ana Pérez Alonso aquí firmantes autorizan a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la propia memoria, como el código, la documentación y/o el prototipo desarrollado.

Madrid, 1 de Julio de 2011

Juan Luis Trincado Alonso	Beatriz Torres Salcedo	Ana Pérez Alonso

Resumen

La visión estereoscópica constituye un campo de la inteligencia artificial que se basa en la composición de imágenes 3D mediante visión binocular. De esta manera, un autómata equipado con un sistema de dos cámaras a la misma altura y desplazadas a cierta distancia, podría reconstruir el entorno que le rodea tridimensionalmente.

El presente proyecto está enfocado a este fin: obtener, a partir de las imágenes, un mapa del entorno que, de la manera más aproximada, represente la distancia a la que se encuentran los objetos en la escena respecto del robot. Con dicha reconstrucción, un robot es capaz de poder detectar y esquivar los obstáculos que se puede encontrar en su camino, pudiendo así planificar distintas rutas seguras que le lleven hasta su objetivo.

Palabras clave: visión estereoscópica, disparidad, estructura 3D, visión por computador, tratamiento de imágenes, inteligencia artificial.

Abstract

Stereoscopic vision is a field of artificial intelligence based on the composition of 3Dimages by binocular vision. In this way, a robot equipped with a system of two cameras at the same height and displaced at a distance among them, could rebuild the tridimensional surrounding environment.

This project is aimed for this purpose: to obtain images from the environment that, as nearly, represents the distance at which objects are located on with respect the robot. With this reconstruction of the scene, the stereovision system in the robot could be able to detect and avoid obstacles that can be found on its path, thus being able to plan different routes that lead to secure targets.

Keywords: stereoscopic vision, disparity, 3D structure, image processing, computer vision, artificial intelligence.

CAPÍTULO 1

INTRODUCCIÓN

1.1. Motivación del proyecto

La visión estereoscópica artificial tiene especial importancia en la robótica. Los robots que son enviados a otros planetas para explorar su superficie, tales como los Mars Exploration Rover, tienen que dirigirse por un terreno accidentado con múltiples obstáculos. Para garantizar que sus movimientos y desplazamientos estén suficientemente garantizados, estos robots hacen uso de la visión binocular, de la misma forma que nosotros los humanos hacemos uso de nuestra visión, por cierto, también binocular.

Nuestros ojos, gracias a su separación relativa, obtienen dos imágenes con pequeñas diferencias entre ellas, esto origina lo que se conoce como disparidad, que será posteriormente analizada. Nuestro cerebro procesa las diferencias entre ambas imágenes y las interpreta de forma que percibimos la sensación de profundidad, lejanía o cercanía de los objetos que nos rodean.

De la misma forma, el robot obtiene dos imágenes con las cuáles puede reconstruir la escena tridimensional (3D) a partir de cada par de imágenes.

Este proyecto está enfocado a este fin: aplicar algoritmos y diferentes técnicas para, a partir de las imágenes, obtener un mapa de disparidad que representa la distancia a la que se encuentran los objetos respecto del robot. La aplicación presentada se ha desarrollado en el lenguaje de programación C#.

1.2. Objetivo del proyecto

El objetivo principal del proyecto consiste en mejorar los resultados obtenidos por el equipo de trabajo que inició el desarrollo de la aplicación durante el curso previo 2009/2010. Se trata pues de la continuación de un proyecto con el que se obtuvieron relativamente buenas representaciones tridimensionales para imágenes estereoscópicas, si bien, en el mencionado trabajo se planteaban las líneas de investigación futuras, ya que los mapas de disparidad podrían mejorarse considerablemente en cuanto a la eliminación de errores surgidos de la propia dificultad del proceso. En este sentido, el presente proyecto constituye por un lado la garantía de verificar que proyectos previos pueden ser continuados, creando así las bases de lo que en el mundo empresarial se considera clave en los desarrollos de proyectos, su continuidad y mantenimiento. Por otro lado, se trata de mejorar un desarrollo previo mediante el diseño y desarrollo de nuevos métodos y algoritmos.

1.3. Organización de la memoria

Esta memoria se organiza como sigue. En el capítulo segundo se analizan los conceptos teóricos en los que está basado el prototipo construido. En el capítulo tercero se detalla el diseño y la arquitectura general del prototipo. En el capítulo cuarto se explican las tecnologías utilizadas para desarrollar el proyecto, así como una explicación de por qué se han elegido. El capítulo quinto contiene un breve manual de usuario para la correcta utilización de la aplicación. El capítulo sexto explica el proceso seguido para desarrollar el proyecto. En el séptimo capítulo se detallan los resultados obtenidos para diferentes imágenes, así como las conclusiones extraídas.

CAPÍTULO 2

CONCEPTOS TEÓRICOS

En este capítulo se introducen algunos de los conceptos teóricos y términos frecuentemente utilizados, así como los algoritmos implementados en nuestro sistema. Nuestras principales fuentes de información han sido las referencias [PAJ07.1] y [PAJ07.2].

2.1. Visión Estereoscópica

El campo de la visión por computador surgió en la década de los años 60 con el objetivo de aproximar las habilidades de la visión humana y la percepción artificial. Más específicamente, la visión por computador incluye tanto la capacidad de adquirir imágenes como la capacidad de analizar y dar sentido a dichas imágenes. Estas imágenes se adquieren a partir de una amplia variedad de dispositivos de captura, son almacenadas digitalmente y procesadas por los diferentes algoritmos diseñados con el propósito de llevar a cabo las tareas específicas más relevantes. La figura 2.1 muestra un ejemplo básico de este proceso.

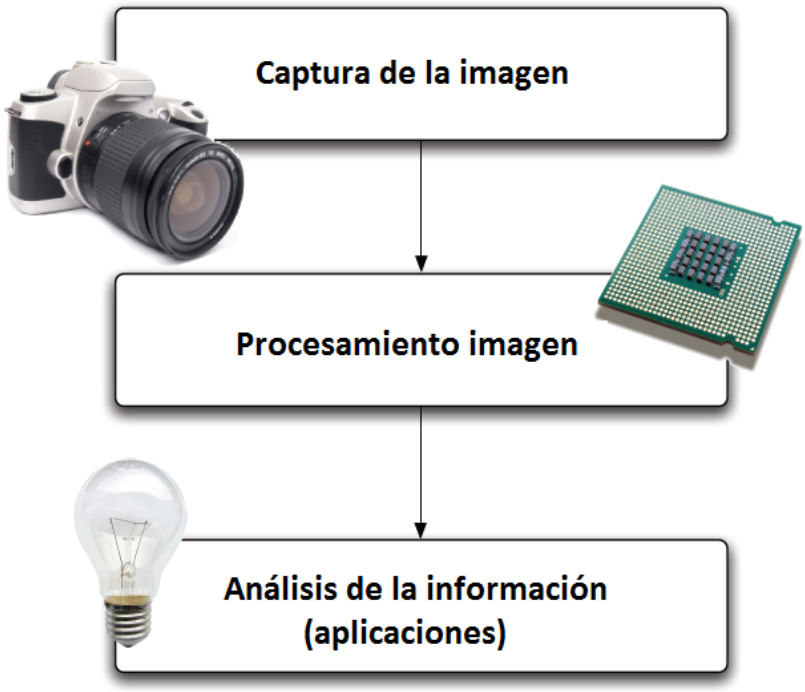


Figura 2.1: Esquema Visión Computador: obtención y procesamiento [LAN07]

En la Visión por Computador, la escena tridimensional 3D es capturada por una, dos o más cámaras para producir imágenes monocromáticas o en color. Las imágenes adquiridas pueden ser segmentadas para obtener de ellas características de interés tales como bordes o regiones. Posteriormente, de esta información se obtienen las propiedades necesarias para reconstruir la escena 3D requerida por la aplicación solicitante. La Visión Estereoscópica constituye un procedimiento más para la obtención de la forma de los objetos en la escena. En este caso la forma se determina a través de la distancia de los objetos en la relación con un sistema de referencia. Se trata de un método más para la obtención de la tercera dimensión de los objetos.

La obtención de la distancia ha sido una preocupación constante en algunos sistemas, particularmente en los diseñados para la navegación en robótica, donde el robot necesita conocer en cada momento la estructura de la escena para navegar en el entorno. Con tal propósito se han desarrollado ciertas técnicas y dispositivos, entre ellos, la que nos ocupa: la visión estereoscópica artificial, donde dos o más cámaras captan la misma escena y por triangulación se determina la distancia.

La visión estereoscópica artificial toma como referencia el modelo estereoscópico biológico. En estos sistemas el desplazamiento relativo de los ojos permite obtener la profundidad de los objetos o tercera dimensión mediante el simple proceso de triangulación a partir de las dos imágenes generadas por el mismo objeto de la escena 3D en cada ojo. Esto es posible porque el hecho de que los ojos estén desplazados entre sí hace que las imágenes de los objetos en sendos ojos se muestren desplazadas según la distancia de los objetos a los ojos.

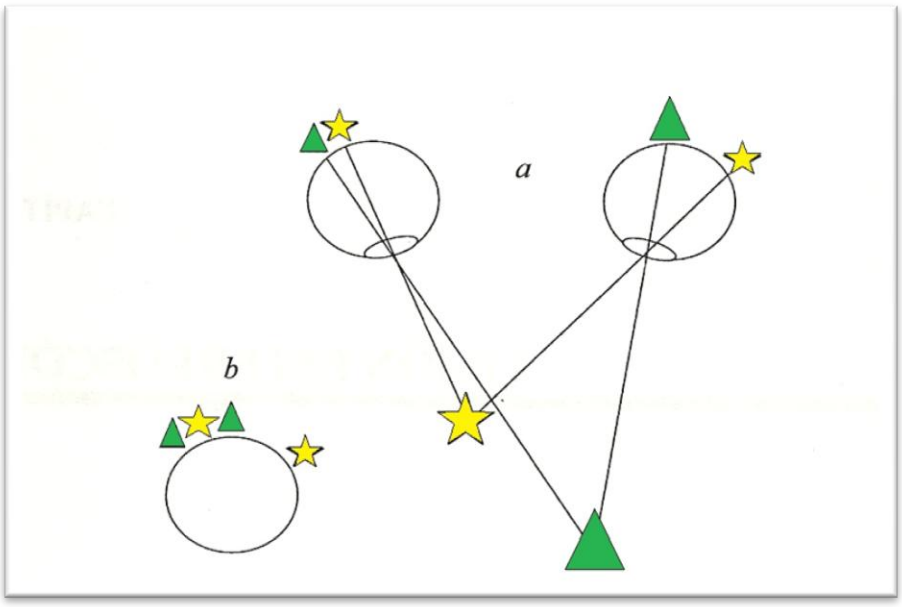


Figura 2.2: Ejemplo gráfico del concepto *disparidad* [PAJ07.1]

Como se puede apreciar en la figura 2.2 (*a*) muestra un sistema estereoscópico biológico observando dos objetos (estrella y triangulo) a distintos niveles de profundidad, en las correspondientes retinas de los dos ojos se obtienen las respectivas imágenes. Si solapamos ambos ojos obtenemos la imagen (*b*) de la Figura 2.2 en la que se observa que la separación relativa entre las imágenes de los dos triángulos es menor que la separación relativa entre las imágenes de las estrellas.

Este fenómeno se explica por el hecho de que la estrella en la escena 3D se encuentra más próxima a los ojos que el triángulo. Estas separaciones relativas entre las imágenes de un mismo objeto de la escena 3D es lo que denominamos **disparidad**.

En la Figura 2.3 se muestran sendas imágenes estereoscópicas, donde puede apreciarse un ligero desplazamiento de los objetos en la escena entre ambas imágenes. Obsérvese, de forma especial cómo las partes de los objetos situadas en los bordes izquierdos y derechos de las imágenes aparecen bajo distintas visibilidades, lo cual es debido al mencionado desplazamiento relativo de las cámaras, siendo a su vez el origen para la obtención de la tercera dimensión o distancia a la que se encuentran los objetos.

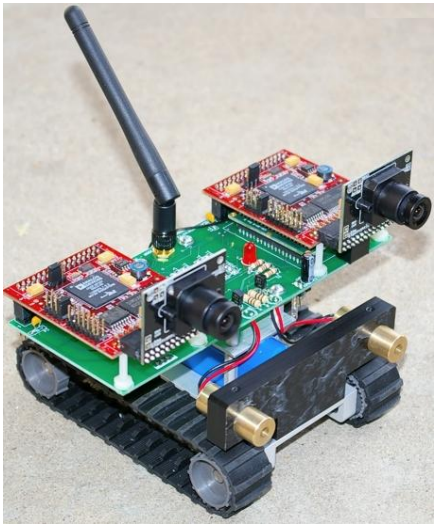
Figura 2.3: Dos imágenes estereoscópicas, izquierda y derecha

En visión estereoscópica artificial se utilizan dos o más cámaras, separadas entre sí a una cierta distancia relativa, con las que se obtienen las correspondientes imágenes. Generalmente se utilizan dos como en el modelo biológico. El procedimiento consiste en captar dos o más imágenes de una misma escena.

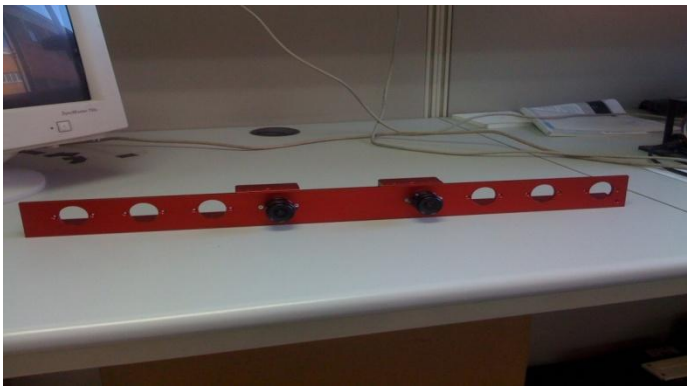
Cada imagen es capturada desde una posición de las cámaras ligeramente diferente a la anterior, de forma que las imágenes se presentan a su vez ligeramente desplazadas entre sí, siendo éste el fundamento básico de la visión estereoscópica, ya que este hecho es el que va a permitir la obtención de la distancia a la que se encuentra un determinado objeto.

La obtención de las imágenes de la escena ligeramente desplazadas se puede obtener por alguno de los dos procedimientos siguientes:

- Alineando dos cámaras de forma que se sitúen ligeramente desplazadas en el espacio. En la Figura 2.4 se muestran varios sistemas estereoscópicos de esta naturaleza instalados en los laboratorios del Grupo de investigación ISCAR (Ingeniería de Sistemas Control, Automática y Robótica) de la Universidad Complutense bajo el que hemos desarrollado nuestro trabajo. En *(a)* aparece el robot Surveyor equipado con dos cámaras de baja resolución que se comunican con el computador mediante tecnología WiFi. En *(b)* el Sistema Videre con posibilidad de situar las cámaras en distintas posiciones. Finalmente, en *(c)* y *(d)* el mismo sistema montado por el mencionado grupo mediante dos cámaras Basler 17FC, que se conectan al computador mediante tecnología FireWire,

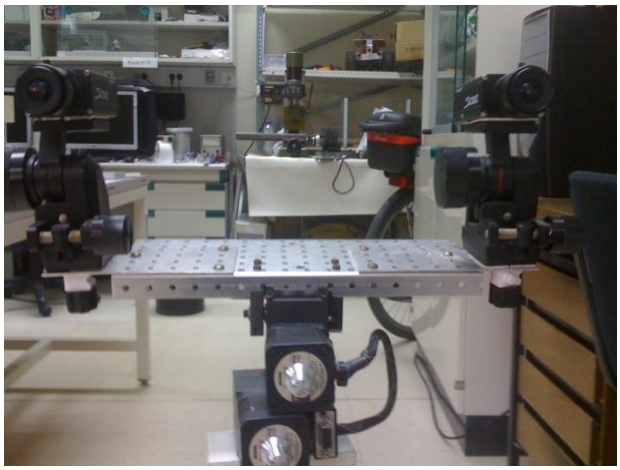


(a)



(b)

(c)



(d)

Figura 2.4: Diferentes Sistemas y equipamientos estereoscópicos: (a) Surveyor con tecnología WiFi; (b) Sistema con línea base variable de VIDERE; (c) y (d) equipamiento estereoscópico de diseño.

- Desplazando una única cámara una cierta distancia y captando las imágenes en las diferentes posiciones de desplazamiento. En la Figura 2.5 se muestra un soporte estereoscópico para captura de imágenes mediante una única cámara, que se desplaza a lo largo de la barra superior, dotada del correspondiente canal de deslizamiento.

Figura 2.5: Soporte estereoscópico para captura de imágenes mediante una cámara

En ambos casos la geometría del sistema puede diseñarse de forma que las cámaras tengan sus ejes ópticos paralelos o convergentes. Gracias a que el sistema humano posee el mejor sistema de visión estereoscópica que se conoce podemos remitirnos a él para verificar muchos conceptos.

Así por ejemplo, cuando intentamos ver objetos lejanos, los ejes ópticos de nuestro sistema visual se sitúan de forma que se pueden considerar paralelos. Por el contrario, cuando queremos ver algo cercano, ponemos en marcha el mecanismo de convergencia de forma que los ejes ópticos lleguen a intersectarse.

En resumen, cuando los ejes ópticos son paralelos, los objetos de interés se sitúan en la lejanía obteniendo una cierta noción de distancia sobre ellos, mientras que la noción de distancia sobre los objetos cercanos queda un tanto difuminada.

Aunque, como se ha mencionado previamente, en el grupo ISCAR se dispone de diferentes sistemas estereoscópicos, debido a la complejidad de los procesos basados en visión estéreo, en este trabajo se ha optado por utilizar imágenes tomadas de la famosa base de datos de Middlebury [MID11] salvo algunas de ellas obtenido con el robot Surveyor.

2.2. Geometría del sistema estereoscópico

Partiremos de un sistema formado por dos cámaras cuyos ejes ópticos son mutuamente paralelos, encontrándose separadas a una distancia que se denomina línea base. Un concepto que surge del hecho de tener más de una cámara es el de *línea epipolar*, tratado a continuación:

Definición: tal y como se observa en la Figura 2.6, sean dos imágenes de un mismo objeto, I_1 e I_2, tomadas por dos cámaras diferentes. Dado un punto, P_1 de I_1, su correspondencia en I_2, P_2, pertenece necesariamente a una línea recta completamente determinada por los puntos P_1 y P_2 que se denomina *línea epipolar*.

En este sistema, los ejes ópticos de las dos cámaras están separados únicamente en una dimensión, la horizontal, por lo que un punto de la escena captado por las dos cámaras va a diferir exclusivamente en su componente horizontal.

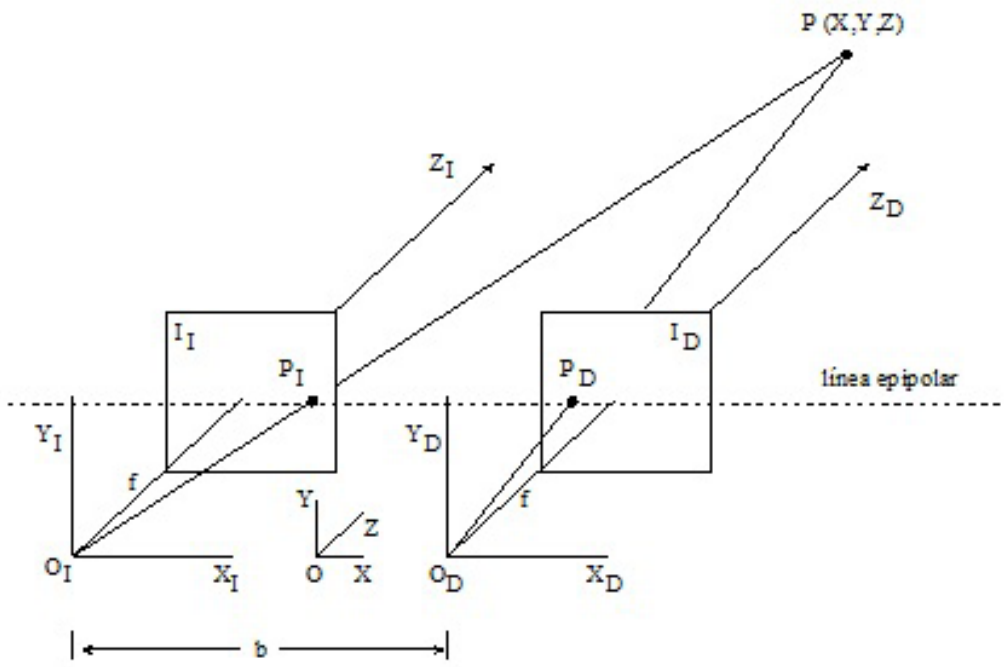


Figura 2.6: Geometría de un par de cámaras con los ejes ópticos paralelos

El origen del sistema de coordenadas es O, siendo la longitud focal efectiva *f*, y la línea base, *b*. En cada una de las dos cámaras se establece un sistema de coordenadas 3D (X, Y, Z), por lo que P_I (X_I, Y_I) y P_D (X_D, Y_D) son las proyecciones en las imágenes izquierda y derecha, respectivamente, de un punto P(X, Y, Z) de la escena.

Los rayos de proyección PO_I y PO_D definen un plano de proyección del punto en la escena 3D que recibe el nombre de *plano epipolar.* Como consecuencia de la geometría del sistema se obtiene la denominada *restricción epipolar,* que limita la búsqueda de correspondencia de manera que en el sistema estándar de ejes paralelos todos los planos epipolares generan líneas horizontales al cortarse con los planos de las imágenes.

En un sistema como el anterior definimos la disparidad de un par de puntos emparejados, esto es, P_I (X_I, Y_I) y P_D (X_D, Y_D) como X_I - X_D.

Utilizando el sistema anterior, podemos obtener, a partir de la geometría del mismo, las ecuaciones 2.1.

$$\left.\begin{array}{l} O_I : \frac{\frac{b}{2}+X}{Z} = \frac{X_I}{f} \\ O_D : \frac{\frac{b}{2}-X}{Z} = \frac{X_D}{f} \end{array}\right\} \Rightarrow \left.\begin{array}{l} X_I = \frac{f}{Z}(X+\frac{b}{2}) \\ X_D = \frac{f}{Z}(X-\frac{b}{2}) \end{array}\right\} \Rightarrow d = X_I - X_D \quad = \frac{f*b}{Z} \Rightarrow Z = \frac{f*b}{d} \qquad (2.1)$$

2.3. Algoritmo de Lankton

El algoritmo de Lankton tiene sus orígenes en un algoritmo de correlación que se basa en el siguiente hecho: Para establecer la correspondencia de un píxel de la imagen izquierda en la imagen derecha se recorre la línea epipolar que pasa por él y se escoge el píxel más similar, entendiendo por similar aquél cuyos valores de intensidad sean más parecidos al que se está tratando de emparejar. Para aumentar la robustez de este proceso, se utiliza un entorno de vecindad alrededor del píxel (ventana).

A continuación nos planteamos como objetivo el estudio para su posterior diseño de uno de los algoritmos estéreo que según el análisis de Middlebury [MID11] ofrece mejores resultados. Este algoritmo, descrito en [LAN07] es una simplificación del propuesto en [KLA06]. Dicho algoritmo aporta como novedad que en lugar de calcular disparidades para píxeles individuales, se calculan planos de disparidad para cada uno de los segmentos de color que se pueden extraer de una de las dos imágenes. La obtención del plano de disparidad óptimo se basa en un algoritmo de propagación de confianza, basada en la obtención de los planos de color.

Un algoritmo de propagación de confianza (en inglés, *belief propagation*) consigue reducir la complejidad computacional de un problema aprovechando la estructura gráfica del mismo. Es por tanto un método de naturaleza bayesiana que se usa especialmente en inteligencia artificial y en teoría de la información.

El primer paso del algoritmo propuesto por Klaus y colaboradores [KLA06] consiste en obtener un plano de segmentos de color a partir de la escena. Una reflexión minuciosa del concepto de disparidad nos lleva a la conclusión de que la disparidad no varía demasiado dentro de cada una de las regiones. Los cambios abruptos de disparidad (y por tanto, de distancia), se producen en los cambios de una región a otra.

La correspondencia entre píxeles es uno de los pasos centrales de cualquier algoritmo de visión en estéreo, tal y como se ha comentado anteriormente. Klaus y col. utilizan como medida de correspondencia entre píxeles una suma ponderada de las diferencias en valor absoluto de intensidad (en adelante haremos referencia a este concepto como CSAD) y una medida basada en el gradiente de la imagen (en adelante, CGRAD). A continuación definimos el concepto de gradiente, que resulta familiar en el contexto de la Física, pero que puede resultar extraño a aquellos lectores que no tengan una base en tratamiento digital de imágenes.

Definición: se define el **gradiente** de una imagen expresada como una función de dos variables, f(x,y), en un punto dado (x,y), como un vector de dos dimensiones, perpendicular al punto donde se calcula, generalmente de borde, y dado por la ecuación 2.2.

$$\vec{G}[f(x,y)] = \begin{pmatrix} G_x \\ G_y \end{pmatrix} = \begin{pmatrix} \frac{\partial f(x,y)}{\partial x} \\ \frac{\partial f(x,y)}{\partial y} \end{pmatrix} \tag{2.2}$$

El vector G posee una magnitud y dirección dadas por:

$$|G| = \sqrt{G_x^2 + G_y^2} \tag{2.3}$$

$$\phi(x,y) = \arctan \frac{G_y}{G_x} \tag{2.4}$$

Una forma de calcular el gradiente expresado en la ecuación 2.2 es aplicar de manera directa la definición matemática de derivada:

$$G_x = \frac{f(x+\Delta x) - f(x-\Delta x)}{2 * \Delta x} \tag{2.5}$$

$$G_y = \frac{f(y+\Delta y) - f(y-\Delta y)}{2 * \Delta y} \tag{2.6}$$

Los parámetros CSAD y CGRAD se definen de la siguiente manera:

$$CSAD(x,y,d) = \sum_{i,j \in N(x,y)} I_1(i,j) - I_2(i+d,j) \tag{2.7}$$

$$CGRAD(x,y,d) = \sum_{i,j \in N_x(x,y)} |\Delta x I_1(i,j) - \Delta x I_2(i+d,j)| + \sum_{i,j \in N_y(x,y)} |\Delta y I_1(i,j) - \Delta y I_2(i+d,j)| \tag{2.8}$$

En la ecuación 2.8, N(x, y) representa una ventana de tamaño 3x3 centrada en el punto de posición (x, y). En caso de que las imágenes sean en color, se realiza la suma sobre cada una de sus tres componentes (rojo, verde y azul).La medida de disparidad pondera los valores de CSAD y CGRAD multiplicando a uno de ellos por un peso w y al otro por su complemento en uno, de esta forma se obtiene:

$$C(x, y, d) = (1 - w) * CSAD(x, y, d) + w * CGRAD(x, y, d) \tag{2.9}$$

Por último, la calidad del resultado se optimiza filtrando aquellos píxeles que no proporcionan una confianza alta. Para ello se realiza el cálculo de disparidades desplazando la imagen izquierda sobre la derecha y viceversa. También se incluye un proceso final de tipo "*winner-take-all*", que viene a sintetizarse como elegir de todos los candidatos posibles el mejor de todos los desplazamientos realizados.

2.4. Algoritmo de Correlación

Este algoritmo estéreo de coincidencia que se propone para su implementación, utiliza segmentación de color en la imagen de referencia y una correspondencia de puntuación que se auto-adapta, optimizando el número de correspondencias (disparidad-color) mayoritarias. La estructura de la escena es modelada por un conjunto de ajustes en el mapa de disparidad con posterioridad. En vez de asignar un valor de disparidad a cada píxel, se asigna a cada segmento un sólo valor de disparidad. La disparidad óptima asignada al plano se aproxima por la aplicación de la propagación de confianza.

El objetivo de este algoritmo es determinar las disparidades que indican la diferencia de localización en los píxeles correspondientes [KLA06].

Este algoritmo está basado en el supuesto de que la estructura de la escena se puede aproximar por un conjunto de planos que no se superponen en el espacio de disparidades siendo cada plano coincidente con al menos un segmento de color homogéneo en la imagen de referencia.

El algoritmo aplica los siguientes pasos o fases:

1. Las regiones de color homogéneo se obtienen mediante la aplicación de una segmentación de color (Algoritmo de Cuantización Vectorial).

2. Se utiliza un método basado en ventanas para determinar las diferencias de distancias en las imágenes (Algoritmo de Lankton).

3. Se asigna un valor de disparidad a una región de color homogéneo tomando como referencia la imagen segmentada por clases de colores.

4. Esta correspondencia del valor de disparidad asociada a la clase de color de la región a la que pertenece el píxel se establece en función del filtro elegido.

El algoritmo de de Cuantización Vectorial es el que se describe a continuación:

El principal objetivo de este método consiste en segmentar la imagen por colores. La segmentación es el proceso por el cual se extrae de la imagen cierta información subyacente para su posterior uso. La segmentación está basada en dos principios fundamentales: discontinuidad y similitud.

Al segmentar una imagen por el principio de similitud obtenemos regiones. Una región es, en líneas generales, un área de la imagen en la que sus píxeles poseen propiedades similares, bien en intensidad, color u otra propiedad.

Para segmentar por clases o patrones de colores, utilizamos un valor umbral T que separe de esta forma los distintos tipos de intensidad de una imagen f(x,y). Cada clase se verá representada por su centro o valor promedio entre todos los píxeles pertenecientes a esa clase.

2.4.1. Descripción del algoritmo Cuantización Vectorial

Este algoritmo asume que el número de clases no coincide inicialmente. Por tanto, se comienza suponiendo una única clase. El algoritmo sigue los siguientes pasos:

1. Para cada **patrón** se calcula su distancia con todos los centros existentes. Para el primer elemento él mismo constituye el primer **centro**.

2. Tomar el centro más cercano utilizando una medida de distancia, por ejemplo, la **euclídea**.

3. Si dicha distancia es menor que un **umbral** determinado previamente, se asocia el elemento a la clase y se calcula la media de todos los elementos que pertenecen a dicha clase. Esta medida nos proporciona el nuevo centro.

4. Si la distancia es mayor que el umbral prefijado se crea una nueva clase, asignando el valor del **centroide** al del elemento.

Por medio de los diferentes algoritmos de correspondencia estereoscópica, bien el método de Lankton o de correlación, se obtienen valores de disparidad que proporcionan la relación existente entre los objetos de la escena y sus distancias al sistema de visión. Los valores de disparidad ordenados convenientemente en forma de imagen proporcionan lo que se conoce como mapa de disparidad. Este mapa de disparidad normalmente contiene errores en sus valores, originados por diversas causas: errores del algoritmo, errores de cómputo, debidos al desalineamiento de las cámaras, diferentes valores de intensidad en las imágenes del par estereoscópico.

Se hace necesario un proceso posterior, con el fin de tratar de corregir o minimizar en la medida de lo posible dichos errores. En este sentido, se orientan las técnicas de filtrado que se describen a continuación.

Los filtros de orden están basados en métodos que utilizan estadísticas de la imagen conocidas como de orden. Estos filtros operan en una vecindad de un determinado píxel, denominada **ventana** y remplazan el valor del píxel central de dicha ventana.

- Filtro mediana: la mediana M de un conjunto de valores ordenados por intensidad es el valor situado en la mitad del conjunto, de forma que la mitad de los valores del conjunto son menores que M y la otra mitad mayores que M. La principal función del filtrado de la mediana es hacer que los puntos con disparidades muy distintas se hagan muy parecidos a sus vecinos, eliminando así los picos de disparidad que aparezcan aislados en el área de vecindad del píxel central.

- Filtro moda: de entre todos los valores en el entorno de vecindad se elige el valor más frecuente, es decir, el valor que más veces aparece.
  Este tipo de filtrado presenta la dificultad de que a veces los valores de intensidad en la vecindad son todos diferentes, con lo cual el valor más frecuente puede ser cualquiera de ellos.

- Filtro media: el filtro de la media funciona mediante la definición de algún tipo de promediado sobre el entorno de vecindad N x N de la ventana. El que implementamos en nuestro sistema es el filtro de la media aritmética, que calcula la media aritmética de los valores de disparidad de los píxeles de la ventana que están dentro de la misma clase que la del centro.

CAPÍTULO 3

DISEÑO TÉCNICO

Una vez establecido el objetivo y elegido el método de correspondencia que ha de aplicarse, se hace necesario definir el diseño bajo el cual se va a proceder a la implementación del mismo.

El análisis y diseño de aplicaciones informáticas debe abordarse con técnicas y metodologías adecuadas, acompañadas por una precisa gestión de proyectos y una eficaz gestión de la calidad.

También es importante poder contar con el soporte de entornos y herramientas adecuadas, que faciliten la tarea del profesional informático y de los usuarios a la hora de desarrollar sistemas de información. Además, una buena gestión de la información es fundamental en las empresas, que constituyen el cauce natural de acceso tras la formación en el campo de la informática. Es por ello por lo que el desarrollo de sistemas de información se ve sometido actualmente a grandes exigencias en cuanto a productividad y calidad, y se hace necesaria la aplicación de un enfoque metodológico en la producción del software, como no puede ser menos en todo proyecto de naturaleza informática.

Para el diseño técnico del sistema, se ha elegido la metodología UML (*Unified Modeling Language*), que permite modelar (analizar y diseñar) sistemas orientados a objetos como el elegido en nuestro caso. Entre los principales motivos del uso del lenguaje UML están el entender, diseñar, mantener y controlar información sobre el sistema a construir. Los conceptos básicos relativos al diseño UML que se han utilizado en este proyecto se encuentran el libro de Arlow y Neustadt,[ARL06]

A partir de la gran variedad de posibilidades que nos ofrece el lenguaje UML, se han elegido los tres tipos de diagramas siguientes, que son suficientemente relevantes y útiles para describir el diseño que se plantea: *diagramas de casos de uso, diagramas de clase* y *diagramas de secuencia*.

Organización del Diseño Técnico

El capítulo que nos ocupa queda dividido en varias secciones. En primer lugar, consta de una visión global del diseño, esto es, las diferentes funcionalidades que ofrece el sistema y las clases principales del mismo. Posteriormente, se desglosa el diseño principal en cada uno de los algoritmos implementados con sus correspondientes diagramas UML que detallan el funcionamiento de cada algoritmo en concreto.

3.1. Diseño Principal

En esta sección se introducen las funcionalidades del sistema representadas en el diagrama de Casos de uso (Figura 3.1) y se presentan las principales clases que constituyen la aplicación.

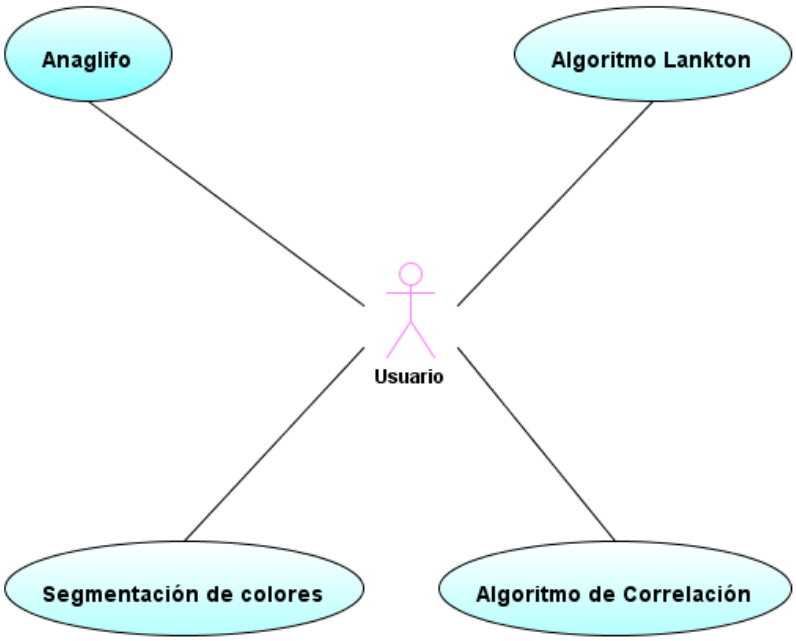


Figura 3.1: Diagrama de caso de uso

Las distintas funciones que soporta nuestro sistema se muestran esquematizadas en el diagrama correspondiente mostrado en la Figura 3.1 y se resumen a continuación:

Anaglifo: el usuario podrá crear un anaglifo, imagen de dos dimensiones capaz de producir una sensación tridimensional cuando se visualiza mediante gafas polarizadas o con filtros Rojo-Azul, Todo ello a partir de dos imágenes 2D de una escena.

Algoritmo Lankton: implementa la funcionalidad de calcular el mapa de disparidad a partir de las imágenes estereoscópicas previamente capturadas por cualquiera de los sistemas de visión estereoscópica mencionados en el capítulo dos. Si bien en la aplicación desarrollada, esas imágenes se obtienen mediante su lectura desde fichero.

Segmentación de Colores: implementa el algoritmo de Cuantización Vectorial para segmentar la imagen por colores.

Algoritmo de Correlación: el usuario podrá mejorar los resultados del Algoritmo de Lankton tomando como referencia las imágenes segmentadas por el Algoritmo de Cuantización Vectorial.

Para describir las clases que constituyen nuestro diseño utilizaremos los diagramas de clase. Éstos son un tipo de diagramas estáticos que describen la estructura de un sistema mostrando sus clases y las relaciones entre éstas. En ellos, aparecen las

operaciones, comúnmente conocidas como métodos, indicando aquellas actividades que se pueden realizar con o para dicho objeto.

Las clases del sistema son en su mayoría interfaces de usuario del tipo .xaml cuyas características se detallan más adelante en el Capítulo cuatro.

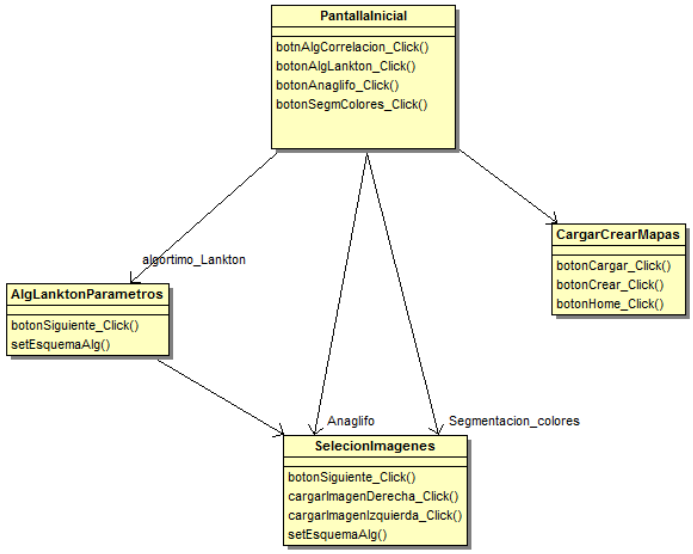


Figura 3.2: Diagrama de clases de Pantalla Inicial

La Figura 3.2 muestra la clase inicial y su relación con el resto de clases con las que establece una relación directa. A partir de la clase *PantallaInicial* podemos acceder a los diferentes algoritmos implementados que se detallan a continuación uno por uno.

3.2. Diseño del Anaglifo

Los anaglifos son imágenes de dos dimensiones capaces de provocar un efecto tridimensional, cuando se visualizan con lentes especiales, esto es con lentes equipadas con filtros de color diferente para cada ojo.

Se basan en el fenómeno de síntesis de la visión binocular. Las imágenes de anaglifo se componen de dos capas de color, superpuestas pero desplazadas ligeramente una respecto a la otra para producir el efecto de profundidad que proporciona precisamente la disparidad existente.

Los pasos a seguir para construir el anaglifo son:

- Para la imagen izquierda, se eliminan los colores azul y verde.
- Para la imagen derecha, se elimina el color rojo.
- Finalmente, la imagen izquierda se desplaza hacia la derecha de forma que ambas queden superpuestas.

De esta manera, la imagen resultante del anaglifo contiene dos imágenes filtradas por color, una para cada ojo. Cuando se ve a través de las gafas especiales, se revelará

una imagen tridimensional gracias a la percepción de la corteza visual de nuestro cerebro.

Esta técnica resulta interesante para comprobar la diferencia de desplazamiento que se produce entre las dos imágenes que conforman el anaglifo. Ése es principalmente el motivo por el cual este algoritmo se incluye en el presente trabajo.

La estructura de clases en que está dividida la parte correspondiente a *Anaglifo* es sencilla. Se compone de tres clases: *SeleccionImagenes*, donde está implementada la interfaz mediante la cual el usuario elige el par de imágenes; *AlgoritmoAnaglifo*, donde está implementado el algoritmo que crea el anaglifo; y *ResultadosAlgoritmo*, donde se muestra el anaglifo construido. Se puede ver en la figura 3.3:

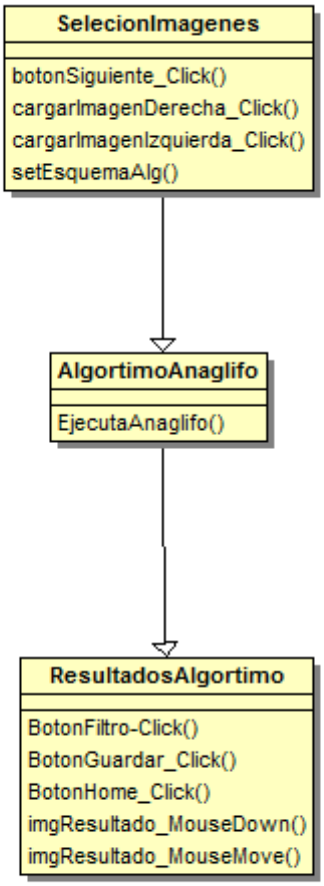


Figura 3.3: Diagrama de clases de *Anaglifo*

La figura 3.4 muestra un ejemplo de ejecución de *Anaglifo*: de la pantalla inicial pasamos a la interfaz de selección de imágenes. Una vez elegidas pulsamos el botón *Siguiente* y accederemos a la interfaz en la cual se muestra el resultado.

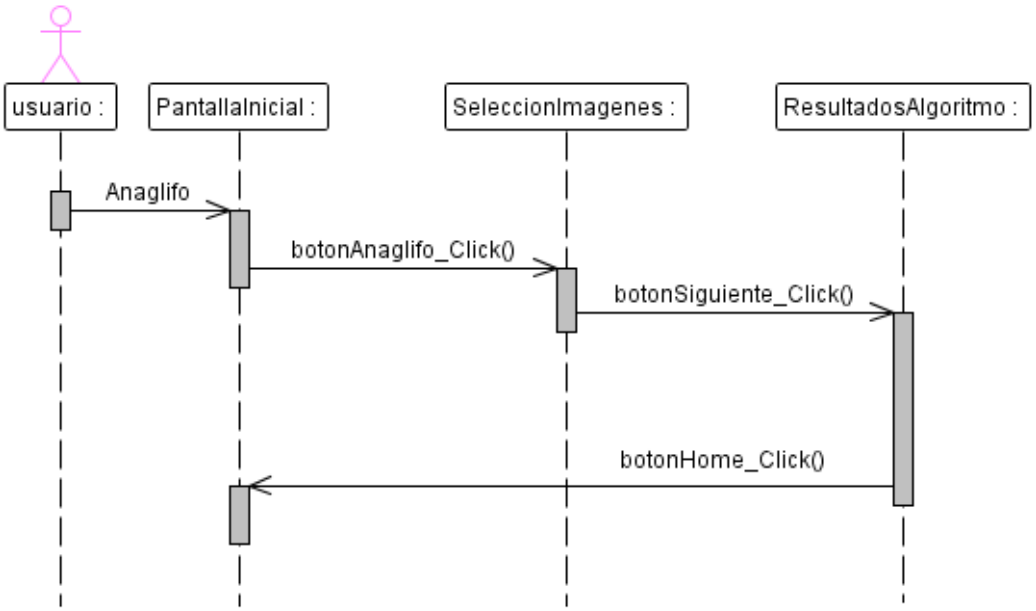


Figura 3.4: Diagrama de secuencia de *Anaglifo*

3.3. Diseño del Algoritmo de Lankton

El algoritmo de Lankton, cuyos fundamentos teóricos se explicaron en el Capítulo dos, es un algoritmo de obtención de matrices de disparidad. Este algoritmo ha sido implementado en el método AlgoritmoLankton.

La idea fundamental de dicho algoritmo es que, tomando dos imágenes, se superpone una sobre la otra y se desliza horizontalmente. Para cada movimiento del deslizamiento se obtiene una matriz de disparidad y se escoge el mejor deslizamiento, entendiendo por tal el que proporcione como resultado una matriz de disparidad con el menor número de píxeles nulos, ya que estos píxeles proceden de decisiones en los que el algoritmo no ha encontrado correspondencia.

Como es posible que las cámaras no se encuentren alineadas, esta operación debe realizarse desplazando también verticalmente algunos píxeles las imágenes (los que el usuario determine). Para cada desplazamiento vertical hay que realizar el deslizamiento horizontal que se comentaba anteriormente.

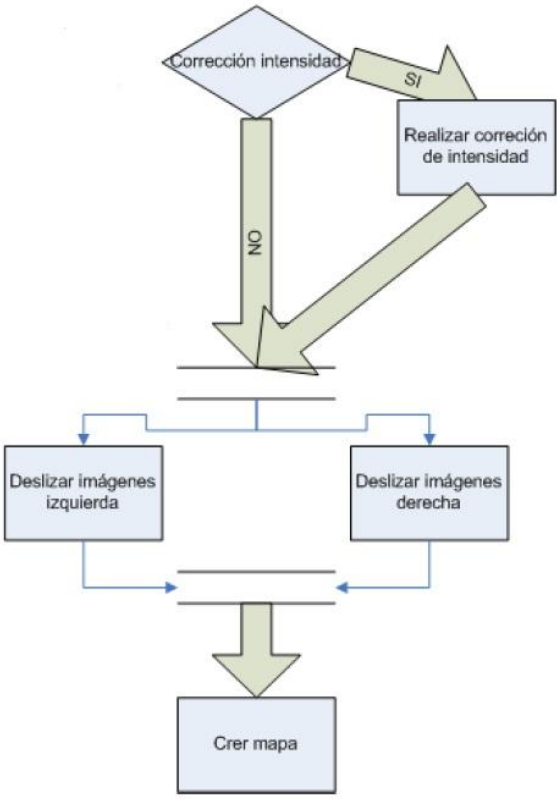


Figura 3.5. Diseño del Algoritmo de Lankton

Como se muestra en la Figura 3.5, el algoritmo comienza con una corrección de intensidad en las imágenes. Es altamente probable, en base a nuestra experiencia, que las dos imágenes del par estereoscópico tengan distintos brillos y niveles de intensidad por lo que es necesario aplicar este filtro corrector de intensidad. Con este filtro se consiguen dos imágenes normalizadas a las que se puede aplicar el algoritmo de Lankton.

El siguiente paso del algoritmo, tomando como referencia la Figura 3.5, sería el desplazamiento horizontal. Este desplazamiento se realiza de las dos formas posibles, esto es, se desliza la imagen derecha sobre la izquierda y al mismo tiempo se desliza la imagen izquierda sobre la derecha. Para agilizar el proceso y mejorar el tiempo de cómputo se utilizan hilos de ejecución. Se realizan sendas copias de la imagen izquierda y derecha para que ambas operaciones se pueda realizar simultáneamente, ya que son procesos independientes.

Por último, se obtienen los mejores valores de disparidad en las matrices de píxeles, y se crea el mapa de disparidades como resultado del algoritmo.

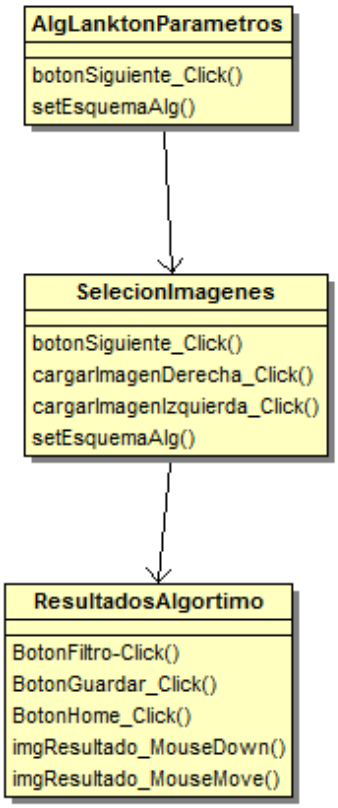


Figura 3.6: Diagrama de clases de Algoritmo de Lankton

El Algoritmo de Lankton sigue el diagrama de clases de la Figura 3.6. Para entender mejor estas clases y su relación entre ellas, describimos la secuencia de acciones que van creando una instancia de cada una de ellas.

Como se puede apreciar en el diagrama de secuencias de la Figura 3.7 el proceso comienza con la acción del usuario que elige ejecutar el Algoritmo de Lankton desde *PantallaInicial*. Para ello se crea una clase *AlgLanktonParametros* que se encarga de almacenar los parámetros especificados por el usuario como son la disparidad máxima, factores de desplazamiento vertical, corrección de intensidad y otros filtros.

La siguiente acción sería la selección de imágenes, izquierda y derecha, y su carga desde la clase *SeleccionImagenes*. Con estas imágenes estereoscópicas representadas en objetos de tipo *Bitmap* (mapas de píxeles) y los parámetros recogidos, se ejecuta en AlagortimoLankton2, que una versión mejorada del Algoritmo de Lankton original, ya que accede de forma paralela a las imágenes para realizar el desplazamiento horizontal.

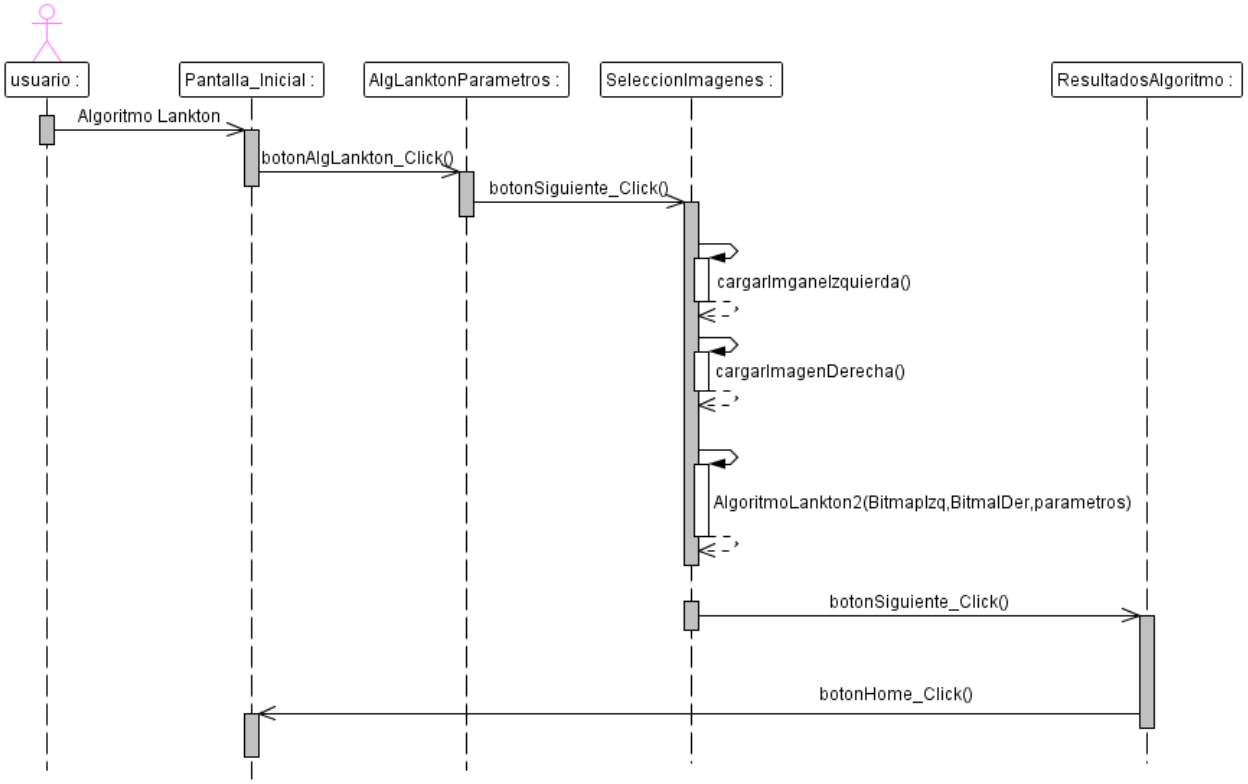


Figura 3.7. Diagrama de secuencias del Algoritmo de Lankton

El resultado generado tras ejecutar este método será un mapa de disparidades que se muestra en la interfaz *ResultadosAlgoritmos.*

Para finalizar, se puede regresar a P*antalla_Inicial* con la función de navegación implementada en el método *botonHome_Click().*

3.4. Diseño de la Segmentación de Colores.

El algoritmo por el cual realizamos la segmentación de colores de la imagen sigue el esquema del Algoritmo de Cuantización Vectorial que se muestra en la Figura 3.8, cuyos fundamentos teóricos se describieron en el Capítulo dos.

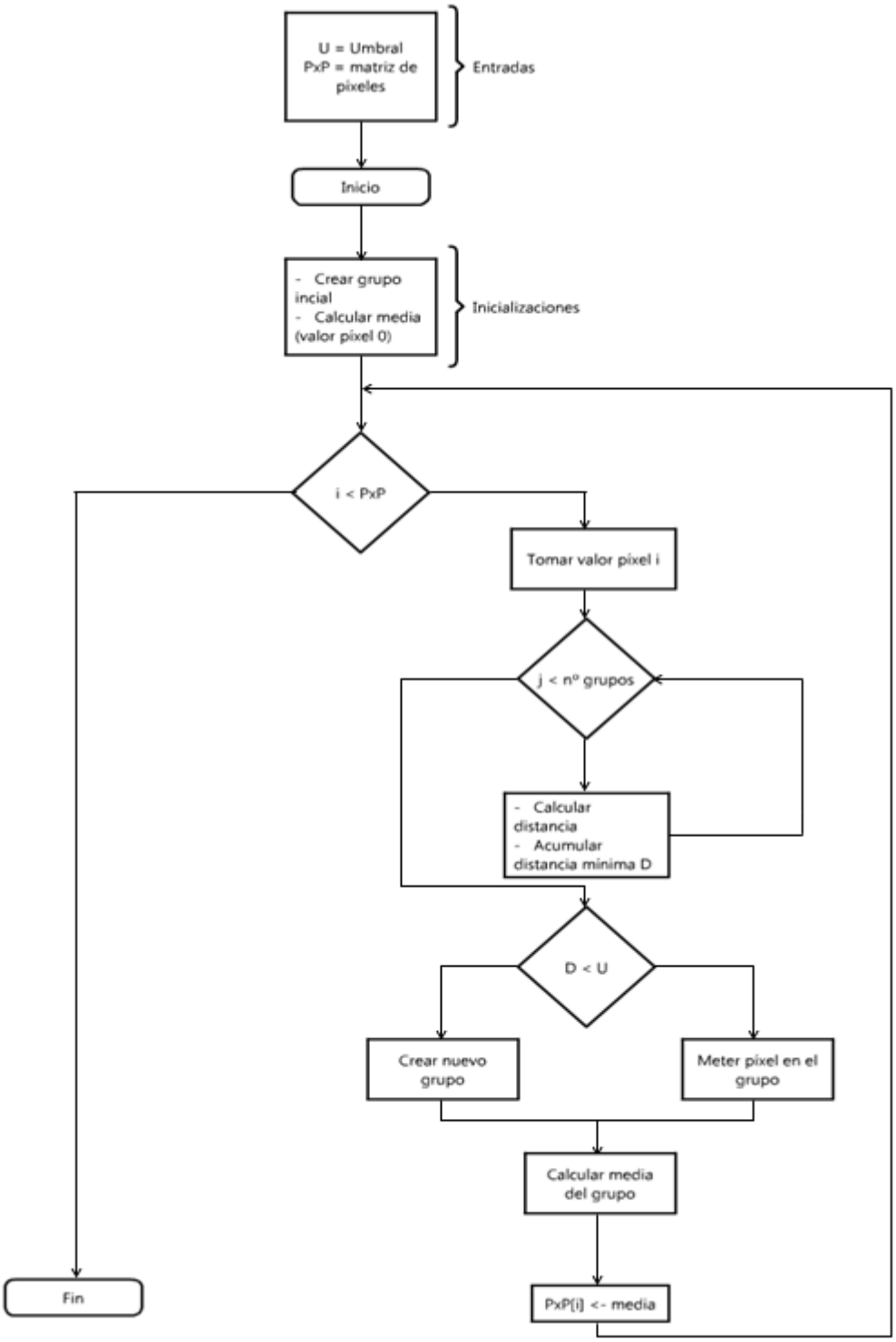


Figura 3.8: Diagrama de flujo del Algoritmo de Cuantización Vectorial

El algoritmo comienza tomando como parámetros de entrada una matriz *M* de píxeles *PxP* y un valor umbral *U* fijado. En función de este parámetro U, el algoritmo establecerá un menor o mayor número de clases o grupos de intensidades.

Este algoritmo comienza con una única clase que se inicializa con el valor del primer píxel de la matriz. Cada clase o grupo de píxeles tiene como propiedad un centro o valor medio de las intensidades de los píxeles que forman esa clase. En el caso de la clase inicial tomará como centro el valor del píxel *M*[0,0].

El siguiente paso del algoritmo es recorrer la matriz de píxeles *M* y agrupar cada píxel en la clase cuyo centro sea más cercano al valor RGB de éste. Para ello, se calcula la **distancia euclídea**, ecuación (3.1)

$$D(\vec{v}_1,\vec{v}_2)=\sqrt{(v_{1,1}-v_{2,1})^2+(v_{1,2}-v_{2,2})^2+(v_{1,3}-v_{2,3})^2} \quad (3.1)$$

Donde $\vec{v}_1=[v_{1,1}v_{1,2}v_{1,3}]^T$ es una tripla de color.

Si el resultado de la distancia entre el píxel *i* y el centro de la clase es menor que el umbral U, dicho píxel i se incluye dentro de esta clase, sino se crea una nueva clase. En ambos casos es necesario calcular de nuevo la media de la clase o centro.

El algoritmo concluye cuando se ha recorrido por completo la matriz de píxeles y el resultado es una imagen segmentada por colores donde cada píxel pertenece a una clase y su valor RGB se ha sustituido por el valor medio de color calculado entre todos los píxeles que pertenecen a su misma clase.

El diseño de la segmentación por colores se basa en el diagrama de clases mostrado en la figura 3.9.

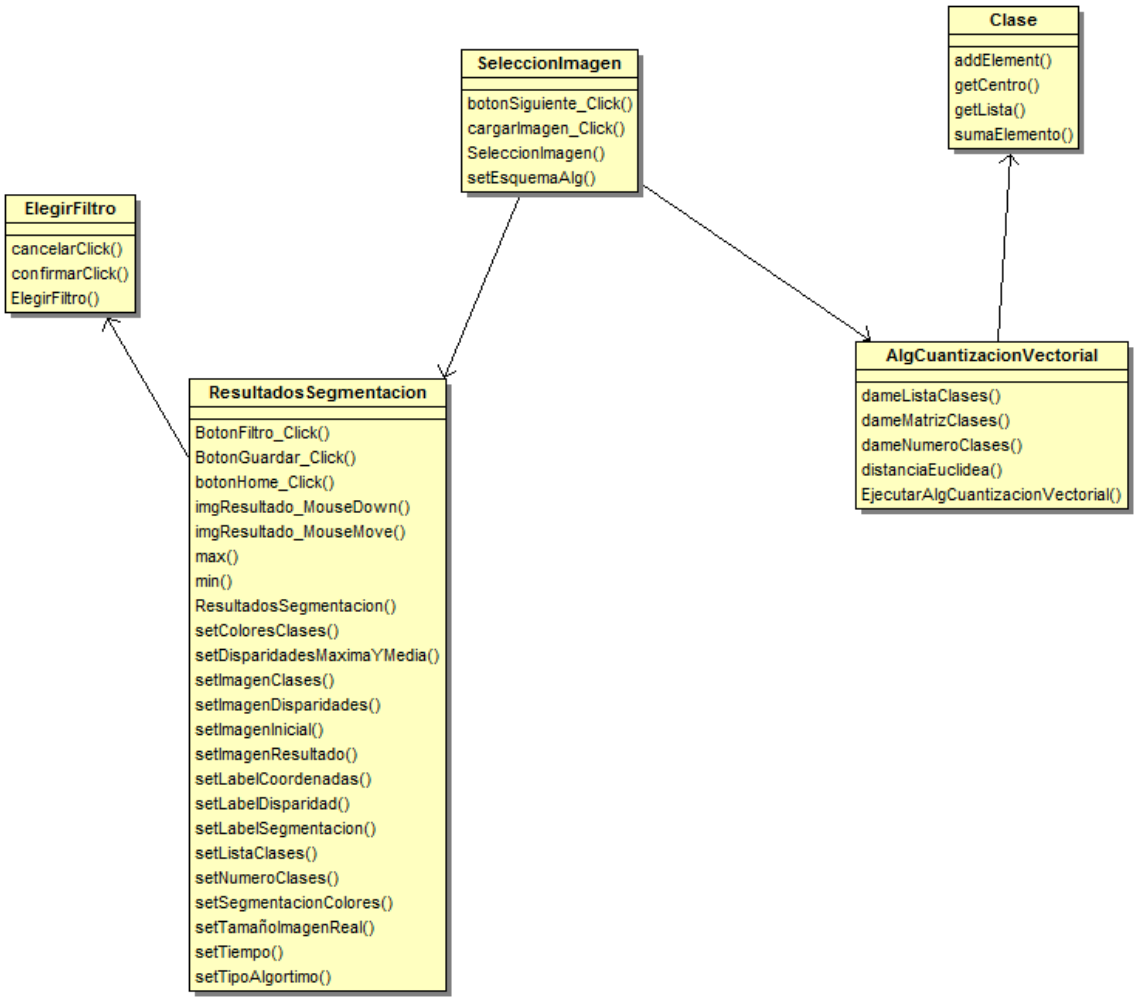


Figura 3.9: Diagrama de clases de Segmentación de Colores

La clase principal donde se inicia el Algoritmo es *SeleccionImagen*, donde se carga la imagen a segmentar. Esta clase está relacionada con *AlgCuantizaciónVectorial*, que es la clase que incluye el método donde se implementa el Algoritmo de Cuantización Vectorial. Ésta a su vez está relacionada con Clase, que es la entidad que representa los grupos de píxeles con valores de color cercanos y cuyo centro es el valor medio de todos estos valores.

La clase *SeleccionImagen*, como se puede observar en el diagrama de la Figura 3.9, está relacionada directamente también con *ResultadosSegmentación*, que es la interfaz donde se mostrará el resultado del algoritmo. Desde ésta existe la opción de aplicar un filtro (clase *ElegirFiltro*) para suavizar los resultados de la segmentación por colores.

Para entender mejor estas clases y su relación entre ellas, describimos la secuencia de acciones que van creando una instancia de cada una de ellas en la Figura 3.10.

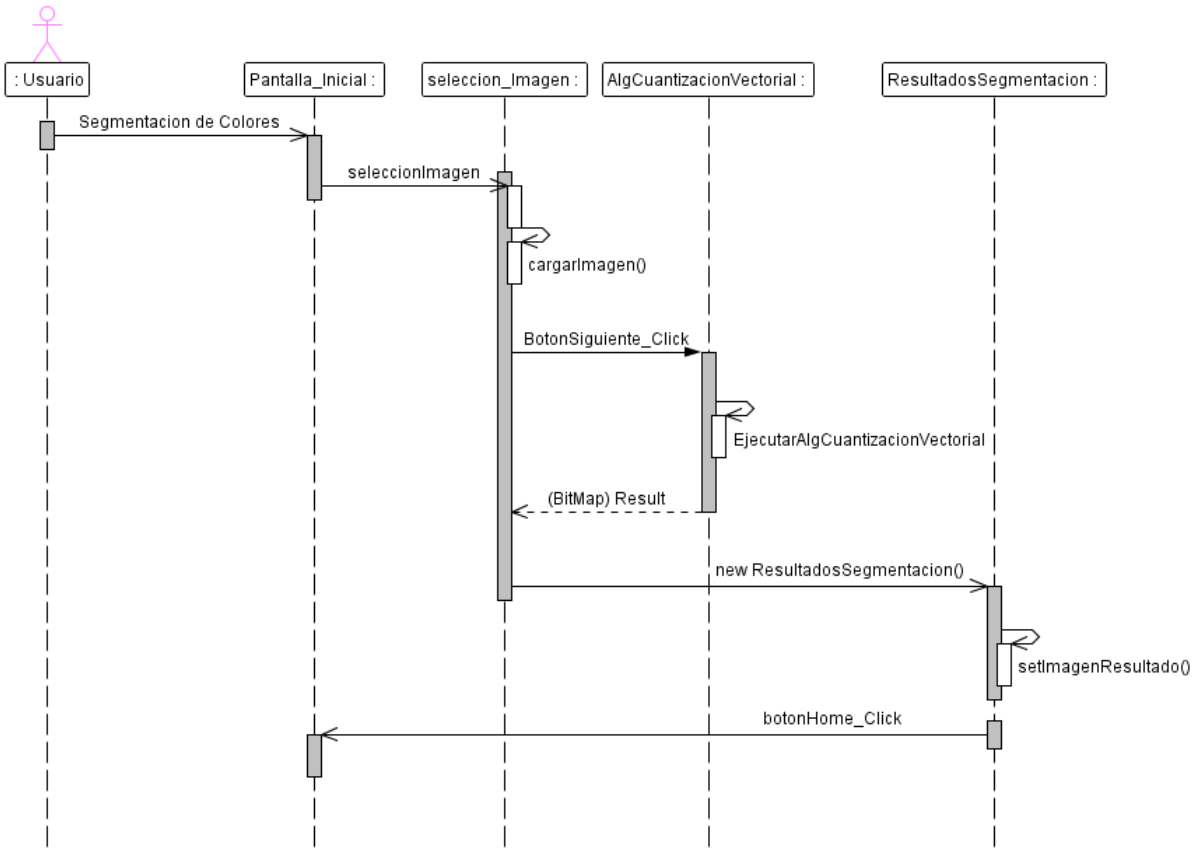


Figura 3.10: Diagrama de secuencia de Segmentación de colores

Como se puede observar en el diagrama de secuencias de la Figura 3.10 el proceso comienza con la acción del usuario que elige ejecutar la Segmentación de Colores desde *PantallaInicial*.

La siguiente acción sería la selección de la imagen a segmentar y su correspondiente carga desde la clase *SeleccionImagen*. Con esta imagen representada como un objeto de tipo *Bitmap* (mapa de píxeles), se ejecuta el método que implementa el algoritmo que nos ocupa, *EjecutarAlgCuantizacionVectorial*. El resultado generado tras ejecutar este método será una imagen segmentada por colores que se muestra en la interfaz *ResultadosSegmentacion.*

Para finalizar, se puede regresar a P*antalla_Inicial* con la función de navegación implementada en el método *botonHome_Click().*

3.5. Diseño del Algoritmo de Correlación

La idea de desarrollar un algoritmo de correlación entre el mapa de disparidad y la imagen segmentada por colores se extrajo del artículo de Andreas Klaus, [KLA06].

Básicamente, consiste en aprovechar el mapa de disparidad que devuelve el algoritmo de Lankton y la imagen segmentada por colores que devuelve el algoritmo de segmentación de colores para corregir aquellos píxeles cuyos valores de disparidad no concuerdan con los que realmente deberían poseer. Su función es corregir dichos errores, que proporcionan información que no se corresponde con la realidad, para proceder a su filtrado y posterior corrección.

En la figura 3.11 se describe la estructura de clases sobre la que se implementa el Algoritmo de Correlación. En *SeleccionImagenesCorrelacion* se encuentra implementada la interfaz de selección de las imágenes a las cuales se les va a aplicar el algoritmo de correlación. *ResultadosAlgoritmo* muestra el resultado final comparando con la imagen de disparidad inicial. En la clase *AlgCorrelacion* está implementado el propio algoritmo de correlación.

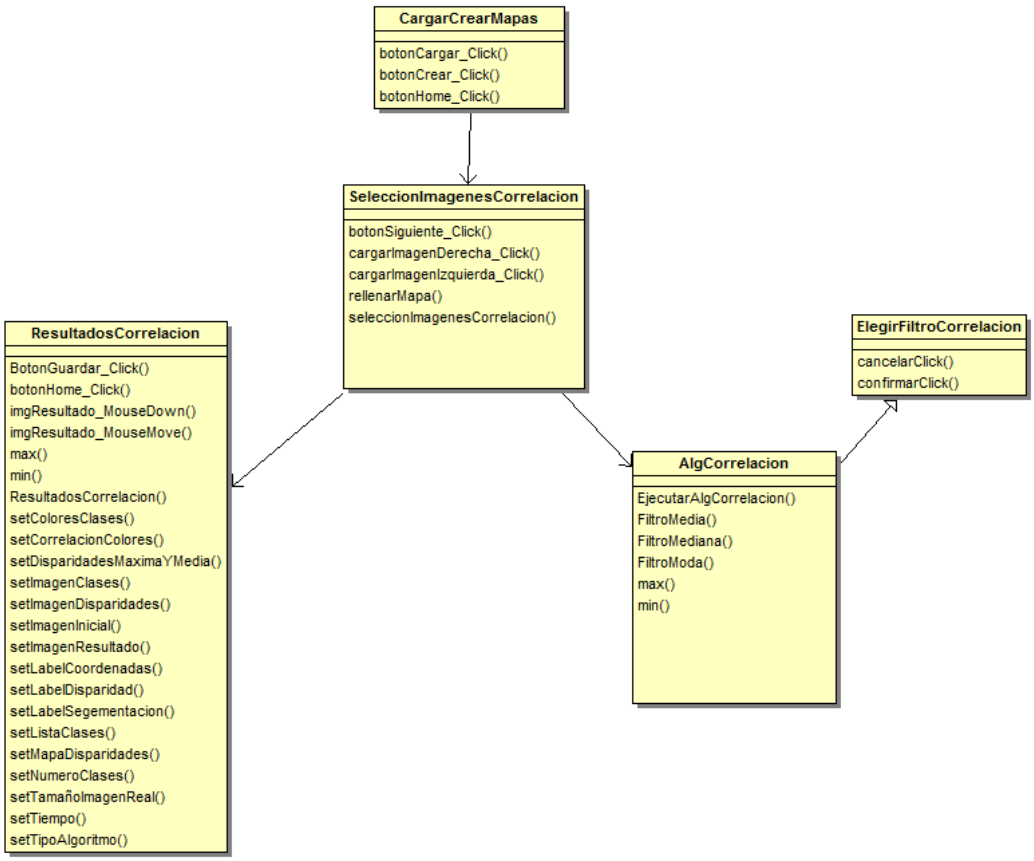


Figura 3.11: Diagrama de clases de Algoritmo de Correlación

A continuación explicamos los diagramas de secuencia principales del algoritmo de correlación recogido en las figuras 3.12 y 3.13:

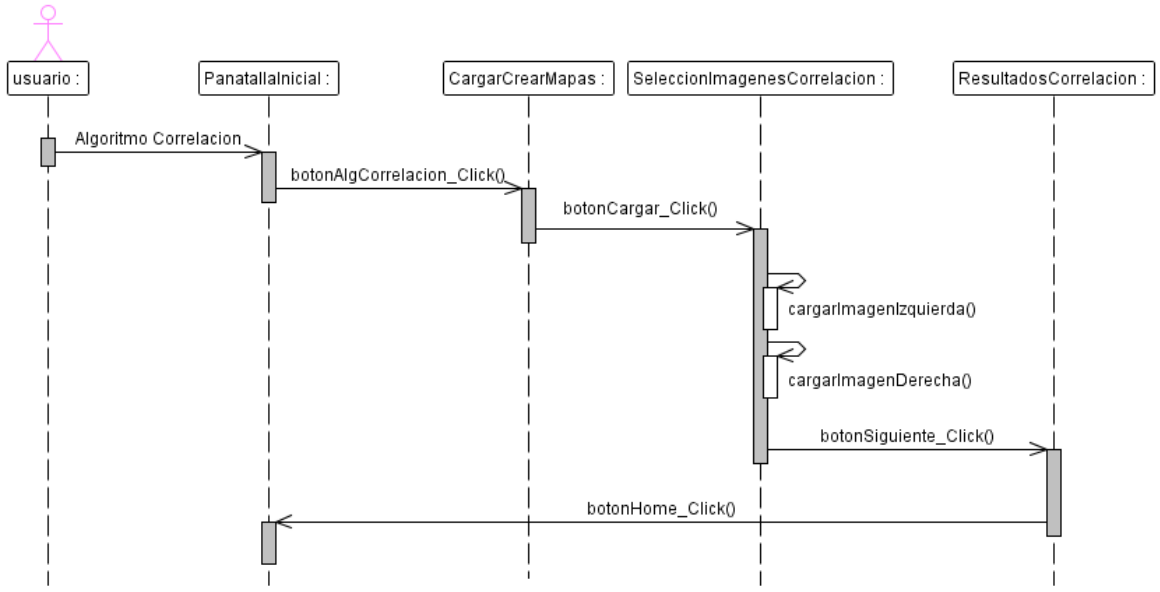


Figura 3.12: Diagrama de secuencia correspondiente a Cargar Mapa

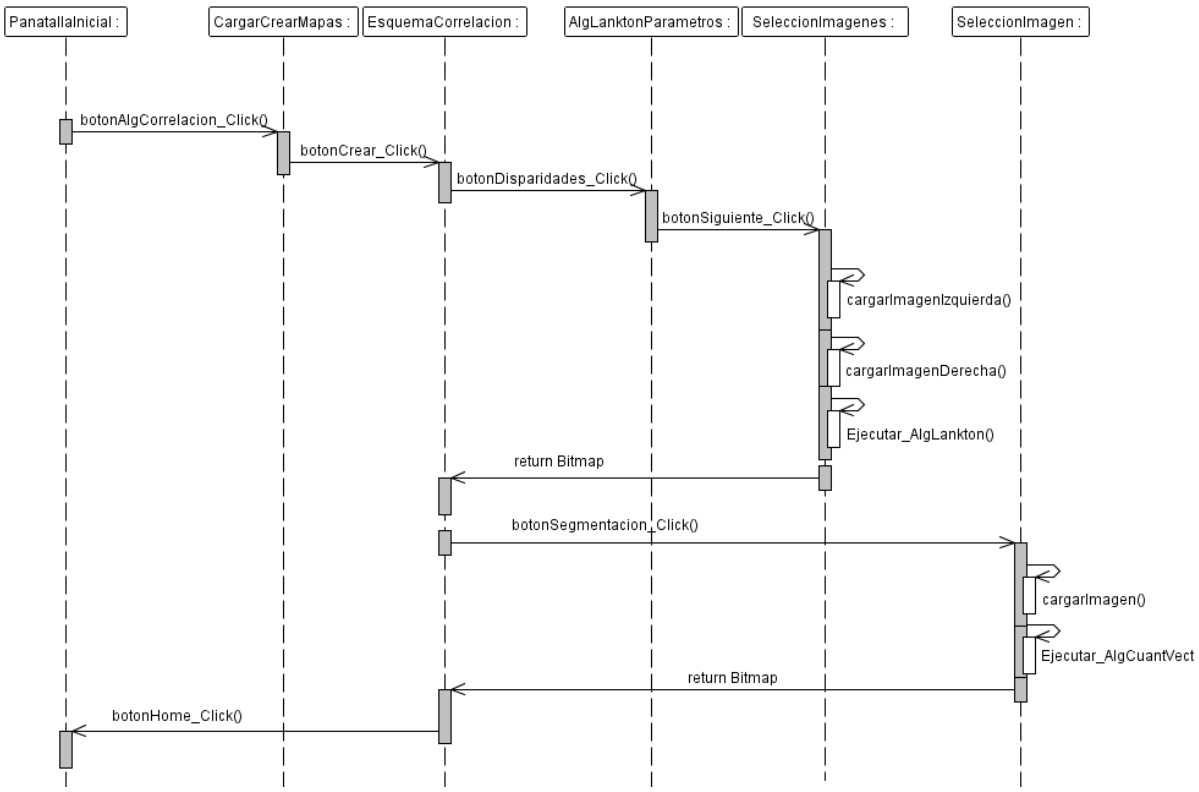


Figura 3.13: Diagrama de secuencia correspondiente a Crear Mapa

En la clase CargarCrearMapas el usuario debe elegir entre *Cargar mapas* o *Crear mapas*, es decir, el usuario puede decidir si quiere aplicar el algoritmo de correlación a un mapa de disparidad y una imagen segmentada ya creados, o crearlos en el momento.

Si elige *Crear mapas* se le redirigirá primero a *Algoritmo de Lankton* y posteriormente a *Segmentación* para crear el mapa de disparidad y la imagen segmentada nuevos (figura 3.12). Si por el contrario elige *Cargar mapas*, el usuario accederá a la interfaz de selección de imágenes de correlación (figura 3.13).

Una vez cargadas las dos imágenes comenzamos a aplicar el algoritmo de correlación, cuyo funcionamiento es el siguiente:

1. Se solicita al usuario un tamaño de ventana impar. La ventana será de tamaño *n x n*, siendo *n* el tamaño de ventana elegido. Usaremos esta ventana para estudiar cuáles son los vecinos de cada píxel, de forma que se pueda comparar con el píxel central de la ventana (de ahí la razón por la cual el tamaño de la ventana es impar).

2. También se solicita al usuario que introduzca un método de filtrado: moda, media o mediana. Este método determina, en función de su nombre, el procedimiento mediante el cual se realiza el proceso de mejora del mapa de disparidad.

3. Recorremos uno a uno todos los píxeles del mapa de disparidad. Para cada píxel, recorremos también uno a uno todos los píxeles de su correspondiente ventana. Para cada posición de esos píxeles, se determina el valor de la disparidad asociada, generando una estructura que contiene todas las correspondencias.

4. Una vez que hemos recorrido todos los píxeles de la ventana, filtramos según haya elegido el usuario. Tenemos tres opciones de filtrado:

    a. Moda: se determina cuál es la correspondencia entre el color que representa la disparidad de cada píxel analizado y la disparidad que más veces ha aparecido dentro de la ventana. Si es mayor que las veces que aparecía la correspondencia entre la disparidad del píxel central y su valor de disparidad, entonces se cambia la disparidad del píxel central por la que más veces había aparecido. En caso contrario se deja como está.
    b. Media: se calcula la media de todos las disparidades que aparecen en la ventana y se cambia la disparidad del píxel central por la de la media de la disparidad
    c. Mediana: se recorre la estructura con las correspondencias, a continuación se van guardando en un “array” todas las disparidades de los píxeles con el mismo valor que el central. Finalizado el recorrido, se cambia la disparidad del píxel central por el de la mediana, es decir, el elemento que ocupa la posición central del ”array”.

5. A continuación nos trasladamos al siguiente píxel y repetimos el proceso desde el paso 3.

6. Cuando se han recorrido todos los píxeles, la aplicación devuelve la representación en colores del nuevo mapa de disparidad obtenido.

Veámoslo más sencillo con un ejemplo: partimos de un mapa de disparidad y una imagen segmentada ya creados (véase figura 3.14):

Mapa de disparidad *Imagen segmentada*

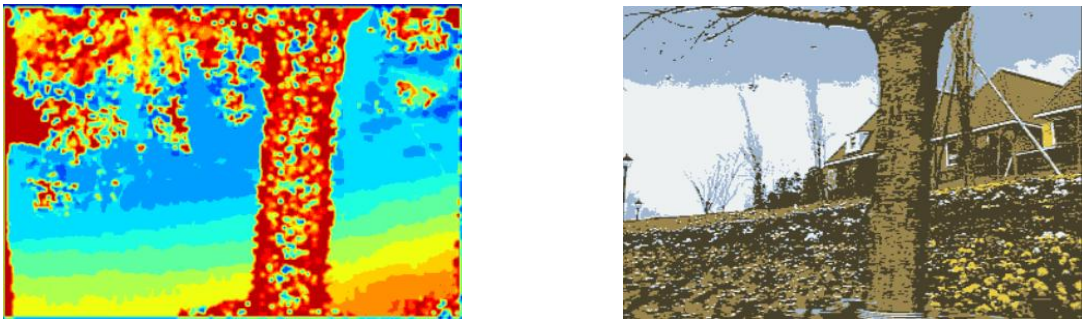


Figura 3.14: Mapa de disparidad e imagen segmentada

Cada una de estas imágenes es la representación en colores de unos valores numéricos de disparidad:

Valores de disparidad

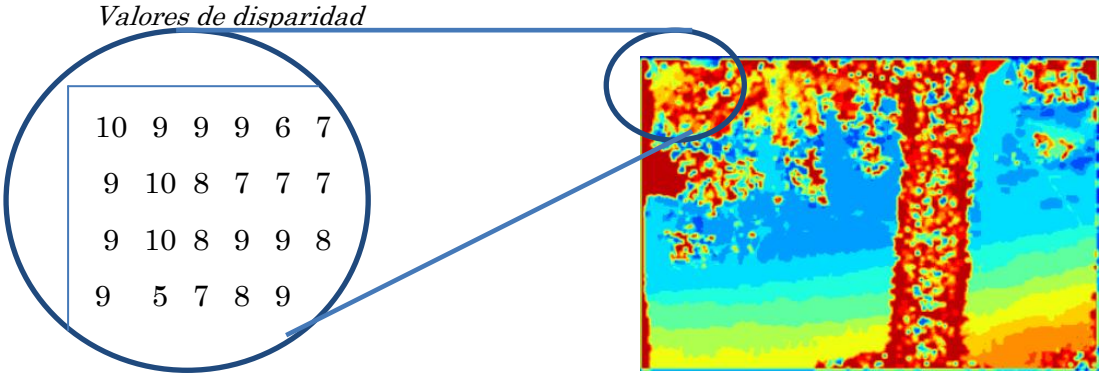


Figura 3.15: Valores de disparidad

Valores de clases de colores

Figura 3.16: Valores de clases de colores

Supongamos que el usuario ha elegido un tamaño de ventana de 3x3, encontrándose en el píxel [2][2] de la matriz correspondiente.

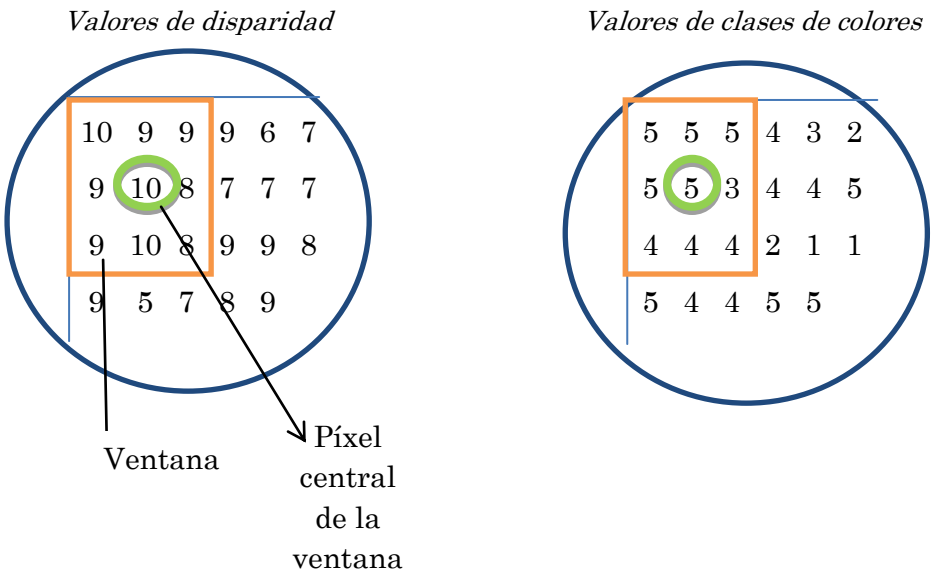


Figura 3.17: Valores de disparidad y de las clases de colores

Si recorremos toda la ventana anotando las correspondencias entre la disparidad y el color, tendríamos una estructura con estas correspondencias:

10 → 5 : 2 veces

9 → 5 : 3 veces

8 → 3 : 1 vez

9 → 4 : 1 vez

10 → 4: 1 vez

8 → 4 : 1 vez

Es decir, el nº de veces que aparece cada vez en la ventana una disparidad con un color.

Supongamos también que estamos aplicando el filtro de la moda. Miramos cuántas veces aparece la correspondencia entre la disparidad del píxel central y su clase de color:

10 → 5 : 2 veces

Ahora buscamos la correspondencia que más veces ha aparecido:

9 → 5 : 3 veces

Como 9 → 5 ha aparecido más veces que 10 → 5, cambiamos la disparidad del píxel central a 9, quedando el mapa de disparidad de la siguiente forma:

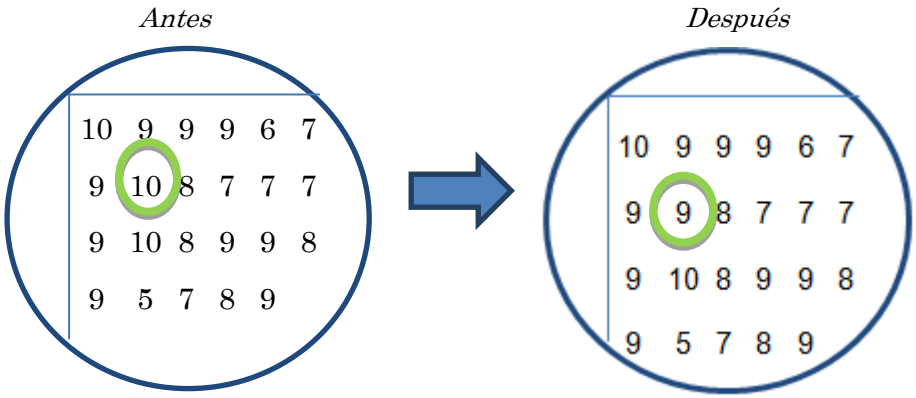


Figura 3.18: Valores de disparidad de antes y después de aplicar el algoritmo

Sólo se modifican los valores de disparidad, el mapa de colores es sólo de lectura.

Una vez hemos acabado con este píxel, avanzamos al siguiente y volvemos a repetir el proceso hasta recorrer todo el mapa.

CAPÍTULO 4

SERVICIOS Y TECNOLOGÍAS UTILIZADOS

En este capítulo se hace alusión a las tecnologías y programas que nos han permitido desarrollar esta aplicación.

4.1. NET Framework

Para desarrollar el proyecto hemos usado el lenguaje de programación C#, que es uno de los lenguajes capaces de utilizar el marco de trabajo .NET de Microsft. C# es un lenguaje de tipo gestionado, como Java, lo que quiere decir que el compilador no genera instrucciones directamente ejecutables por el procesador. En su lugar genera un código en lenguaje intermedio denominado CIL (Common Intermediate Language) que se convierte al código nativo en tiempo de ejecución por un componente denominado compilador JIT (Just In Time).

El entorno de programación (IDE) que hemos usado para escribir códigos C# ha sido la edición Professional de Visual Studio 2010. Visual Studio es un entorno similar al resto que se utilizan en otros lenguajes de programación (Eclipse, NetBeans), etc.). Ofrece el concepto de soluciones y de proyectos con el objetivo de organizar el código fuente y abstraer el proceso de compilación, que se realiza mediante el compilador C# de Microsoft, Csc.exe. También admite multitud de complementos o plug-ins, como el AnkhSVN el cual hemos usado para poder integrar un repositorio de código fuente en el desarrollo de nuestro proyecto con el fin de que cada miembro del grupo pudiera contribuir de manera eficiente.

El lenguaje C# puede resultar muy familiar para aquellas personas con un cierto manejo del lenguaje Java, el cual constituye una materia esencial en la práctica totalidad de las Escuelas de Informática.

.NET Framework está compuesto por un conjunto de ensamblados (DLL) que engloban las diferentes funcionalidades más comunes requeridas por los proyectos de programación: Entrada/Salida, interfaces gráficas, estructuras de datos comunes, sincronismo y paso de mensajes, entre otras. Concretamente, para nuestro proyecto nos hemos centrado en las capacidades de tratamiento gráfico de .NET. El ensamblado System.Drawing proporciona diversas clases que abstraen algunos conceptos de tratamiento y representación de imágenes digitales. La clase Bitmap es la que representa una imagen de mapas de bits en .NET Framework. La constructora de esta clase recibe como parámetro la imagen que queramos representar en memoria y ésta queda almacenada básicamente como una matriz de bits que representan una terna de

colores rojo, verde y azul. Dicha clase dispone de propiedades accesorias y montadoras para acceder o modificar cada uno de sus píxeles.

Dentro de todas las librerías que ofrece .NET Framework, nuestra aplicación hace uso de AForge.NET. Se trata de un proyecto dirigido a investigadores en las áreas de visión por computadora, robótica e inteligencia artificial. Contiene una interfaz para el tratamiento de imágenes dentro del ensamblado AForge.Imaging.Filters. Nosotros hemos hecho uso de este ensamblado nuestro proyecto.

4.2. Windows Presentation Foundation (WPF)

Todo proyecto software además de cumplir con una serie de requisitos en cuanto a su calidad, debe ser amigable y en cierta medida vistoso para el usuario. Es por ello que hemos hecho uso de una tecnología de Microsoft para crear interfaces gráficas de última generación, Windows Presentation Foundation (WPF).

WPF introduce XAML (Zamel), un lenguaje basado en XML que nos permitirá definir el diseño gráfico de nuestros formularios. Esta herramienta está orientada tanto a desarrolladores como a diseñadores, mostrando, si así se desea, en el momento de crear el formulario, el aspecto externo de tanto de la interfaz cómo su implementación en XAML. De esta forma, podemos realizar interfaces gráficas de manera sencilla e intuitiva en poco tiempo, pudiendo modificar los atributos de los objetos creados en cualquier momento mediante la API del programa o yendo directamente al código. En la figura 4.1 se puede ver un ejemplo de uso de WPF con una interfaz de prueba, en la cual se ha insertado un botón y una etiqueta con un texto: en el lado izquierdo aparece el aspecto que presentará la interfaz al usuario de la aplicación y en el lado derecho la implementación de dicha interfaz en lenguaje XAML.

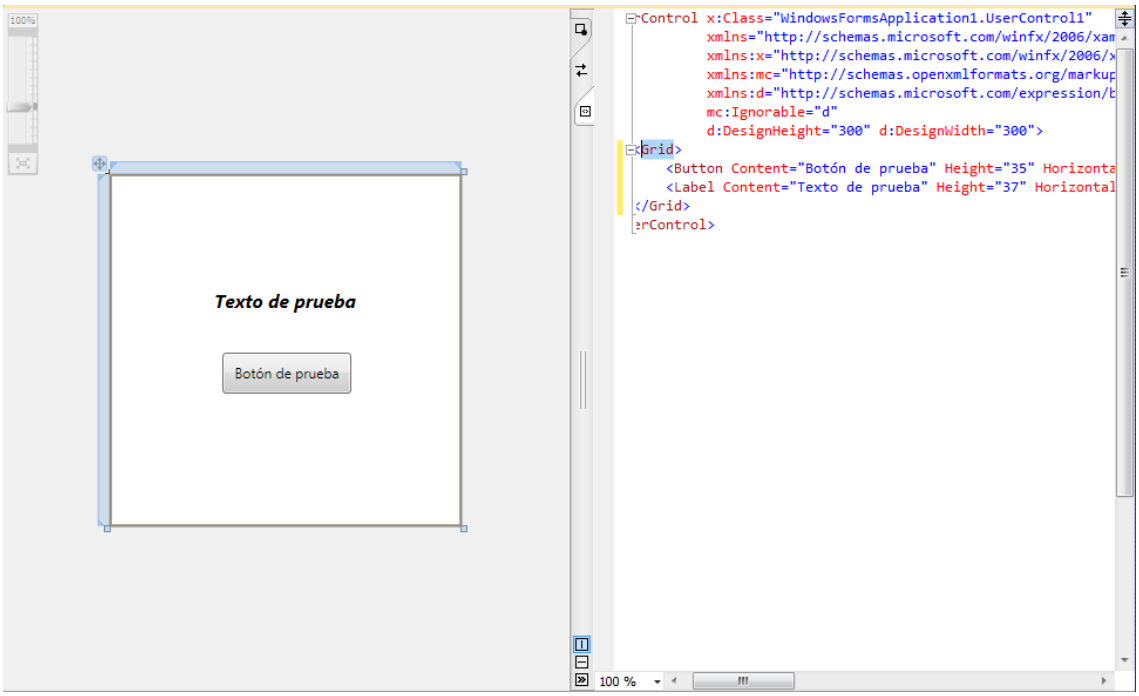


Figura 4.1: Ejemplo de aplicación WPF

CAPÍTULO 5

MANUAL DE USUARIO

La interfaz de usuario ha sido diseñada de la manera más intuitiva posible para los usuarios de la aplicación, presentando una ventana de entrada tal y como se muestra en la Figura 5.1.

Está constituida por una serie de botones que nos permiten desplazarnos por las distintas posibilidades de la aplicación. Además, en la parte superior izquierda de la interfaz tenemos un par de botones de retroceso y avance para poder navegar de forma más cómoda, y que en caso de cometer un error, podamos retroceder fácilmente para cambiar lo que necesitemos sin necesidad de empezar de nuevo desde el principio.

En la ventana principal podemos contemplar los distintos botones, de forma que según se elija uno u otro, se aplicará el algoritmo que le corresponde.

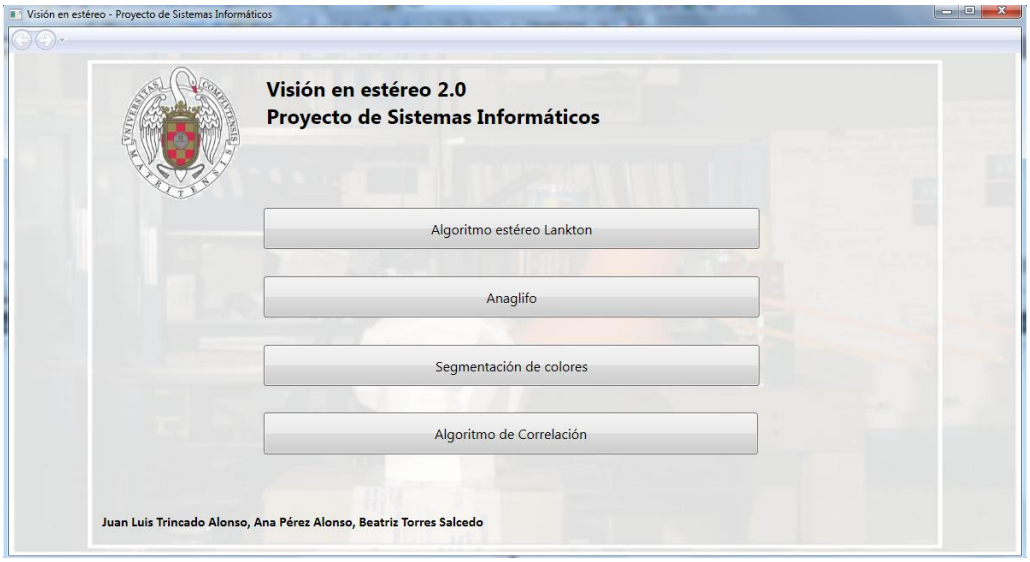


Figura 5.1: Pantalla principal de la aplicación

Como puede verse en la propia Figura 5.1, se dispone de los siguientes botones:

- Algoritmo estéreo Lankton
- Anaglifo
- Segmentación de colores
- Algoritmo de correlación

5.1. Algoritmo estéreo de Lankton

La primera opción aplica, sobre un par de imágenes elegidas por el usuario, el algoritmo de Lankton, a quien debe su nombre. Este algoritmo se explicó en el Capítulo dos relativo a los conceptos teóricos.

En la pantalla principal elegimos Algoritmo estéreo Lankton. Tras pulsar dicho botón, nos aparecerá una nueva ventana de parámetros, tal como se muestra en la Figura 5.2.

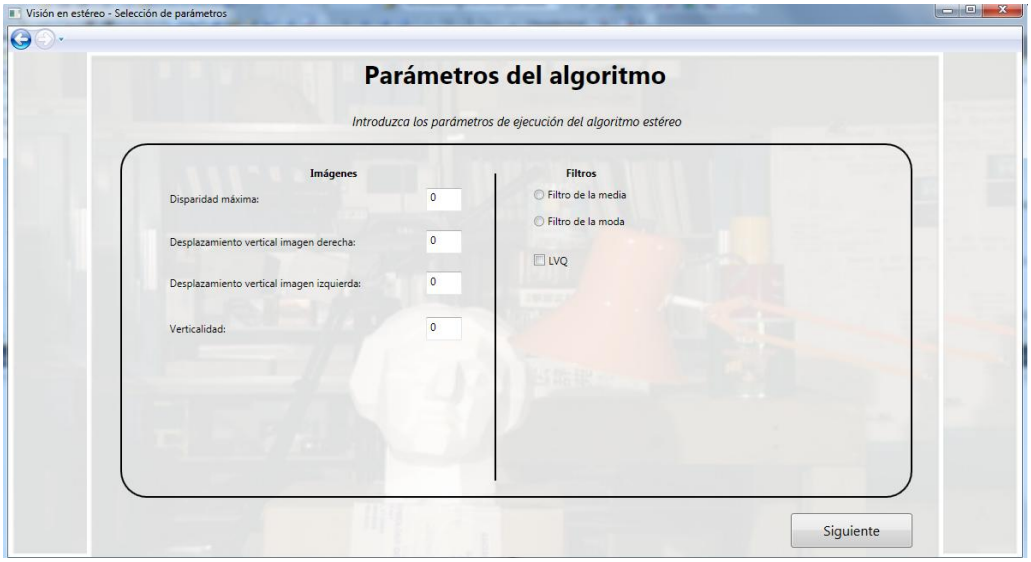


Figura 5.2: Pantalla de parámetros del algoritmo Lankton

Entre los parámetros que podemos encontrar son:

1. **Disparidad máxima**: Es la distancia en número de píxeles entre un mismo punto en la imagen izquierda y la derecha. Realmente, lo que queremos calcular es la disparidad en cada punto, por lo que, en función del tamaño de la imagen y de las cámaras, tenemos que acotar la búsqueda que realizará el algoritmo indicando el valor máximo de disparidad que se detectará para ahorrar tiempo de ejecución.

2. **Desplazamiento vertical:** Los desplazamientos verticales de las imágenes funcionan de la siguiente manera: cuando se introduce un valor positivo, la imagen que hayamos desplazado se moverá hacia abajo, y los píxeles sobrantes en la parte superior se rellenarán de ceros, lo que es equivalente a añadir píxeles negros.

3. **Verticalidad:** Se utiliza para imágenes que no están exactamente alineadas en la dirección horizontal, para que de esta forma, se examinen píxeles superiores e inferiores en ambas imágenes de cara a encontrar la mejor correspondencia. Normalmente su valor por defecto será cero (imágenes alineadas). Cualquier incremento en los valores de este parámetro tendrá como consecuencia un aumento considerable del tiempo de ejecución.

En la parte derecha de la Figura 5.2 aparecen dos parámetros más:

1. **Filtros:** Se permite al usuario elegir entre dos posibles filtros. Ambos sirven para eliminar la excesiva variedad de píxeles que se producen al calcular las disparidades ya que esto proporciona un excesivo número de píxeles con disparidades muy cambiantes, haciendo además que su visualización sea muy difusa.

   Estos filtros se aplican sobre la ventana del tamaño elegido.

   - Media: En el caso del filtro de la *media*, se trata de tomar todos los valores de la ventana, calcular su media (se suman todos sus valores y se divide entre el número de casillas). Tras lo cual se sustituye el valor del píxel central por el valor calculado.

   - Moda: En el caso del filtro de la moda calcula, como su propio nombre indica, la moda de los valores de la ventana, es decir, el valor de disparidad que más se repite en la misma. Al píxel central de la ventana se le asigna el valor de la moda, así obtenido.

2. **LVQ:** Esta opción sirve para aplicar el filtro de normalización de intensidad. Es muy útil para aquellas situaciones en las que la luminosidad de ambas imágenes del par estereoscópico es distinta debida a reflejos de luz producidos por el Sol o por fuentes de luz artificial. El proceso por el cual se realiza esta corrección de la intensidad se basa en el algoritmo de clasificación *Learning Vector Quantization* (LVQ).

Después de elegir los parámetros según las necesidades del usuario, presionamos el botón *Siguiente*. Recordamos que siempre que se desee se puede retroceder con los botones de navegación situados en la parte superior izquierda de la interfaz.

Tras elegir los parámetros deseados y pulsar el botón Siguiente, llegaremos a la pantalla de carga de imágenes. Elegiremos las dos imágenes a procesar y procederemos a ejecutar el algoritmo. En la interfaz gráfica se visualizan las dos imágenes seleccionadas, Figura 5.3.

Figura 5.3: Pantalla de selección de imágenes con dos imágenes ya cargadas.

Aquí se nos permite elegir las imágenes que deseemos para posteriormente ejecutar el algoritmo. Es fundamental situar en la parte izquierda la imagen izquierda y en la parte derecha la imagen derecha del par estereoscópico, ya que de lo contrario el algoritmo no obtendrá los resultados previstos al esperar esta configuración. Las imágenes se podrán cargar usando los botones de *Cargar imagen...*, que permiten buscar en nuestras carpetas las imágenes deseadas.

Una vez elegidas las imágenes que deseemos, podemos proceder a ejecutar el algoritmo teniendo como base dichas imágenes y los parámetros fijados. Pulsamos el botón *Ejecutar*, y tras una espera de tiempo variable obtendremos los resultados esperados.

El resultado se presentará con una imagen que muestra las disparidades coloreadas en función de la distancia de los distintos elementos que componen la imagen. En la Figura 5.4 podemos ver el resultado obtenido como consecuencia del cálculo de disparidades de dos imágenes tomadas en un entorno de interior en el que el objeto más significativo está representado por un balón.

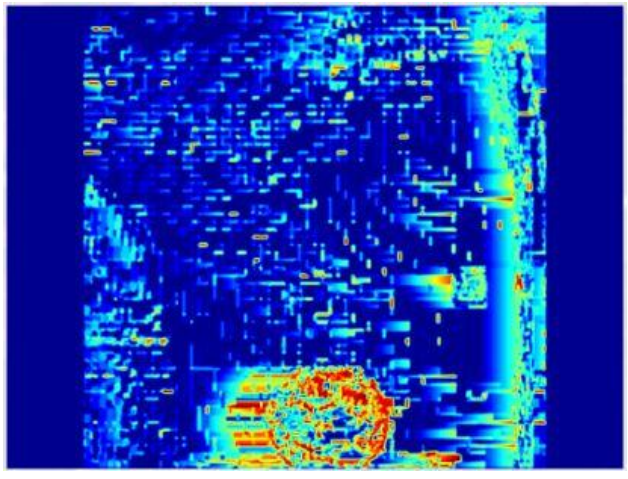


Figura 5.4: Pantalla de resultados del algoritmo Lankton.

5.2. Anaglifo

Regresamos a la pantalla principal pulsando de nuevo el botón *Ir al inicio* y elegimos la siguiente de las opciones: *Anaglifo*.

Los detalles sobre el concepto de anaglifo y cómo se construye a partir de dos imágenes estereoscópicas se describe en el Capítulo tres relativo al diseño técnico de los algoritmos.

Para crear un anaglifo pulsamos el botón *Anaglifo* que nos conducirá de nuevo a una pantalla donde seleccionaremos la imagen izquierda y la imagen derecha.

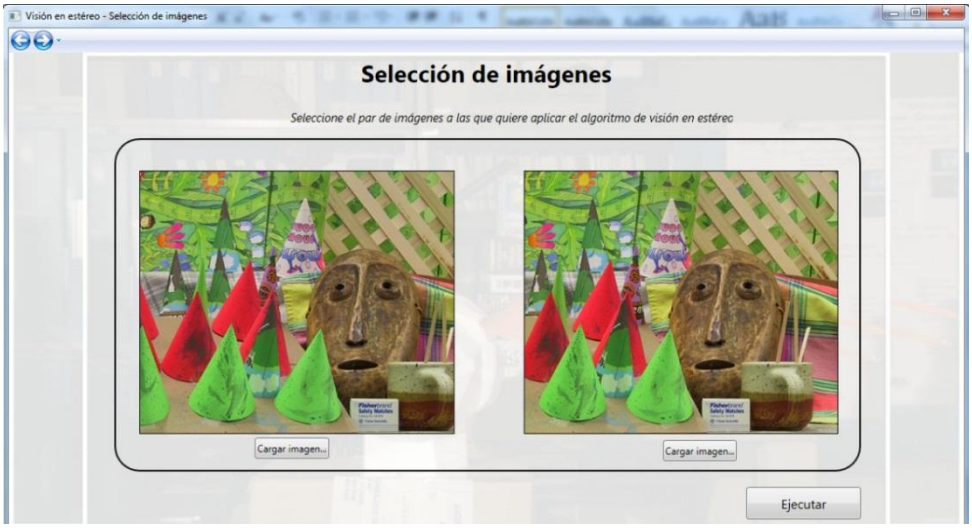


Figura 5.5: Imágenes seleccionadas para realizar el anaglifo

Esta vez hemos elegido otras dos imágenes distintas (figura 5.5), el resultado obtenido es nuevamente el anaglifo, figura 5.6.

Figura 5.6: Resultados al realizar el anaglifo

5.3. Segmentación de Colores

El siguiente botón que nos encontramos es *Segmentación de Colores*. Este algoritmo explicado ya en el Capítulo tres implica la detección, mediante una serie de procedimientos (Algoritmo de Cuantización vectorial), de los contornos o regiones de la imagen.

Pulsamos el botón, y al contrario que el resto de los botones explicados anteriormente, nos lleva directamente a la selección de una única imagen.

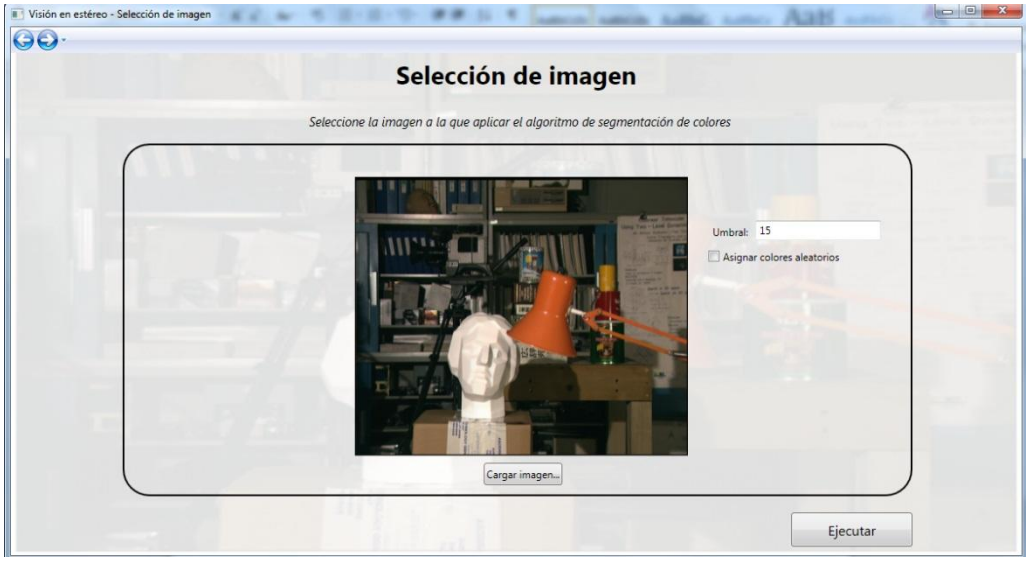


Figura 5.7: Selección de imagen de segmentación.

Seleccionamos la imagen que queramos segmentar y elegimos el umbral.

1. **Umbral:** Separa los distintos tipos de intensidad de una imagen. Cuanto más bajo sea el valor del umbral, mayor será el número de clases obtenidas.

Tenemos la opción de elegir:

2. **Asignar colores aleatorios:** Asignará a la imagen resultado colores aleatorios.

Ejecutamos el algoritmo y obtenemos la imagen resultado. Tras su ejecución se obtiene la siguiente información.

- Los colores con los que se ha realizado la segmentación de colores
- El número de colores
- El tiempo que ha tardado en ejecutarse el algoritmo

Además si pulsamos sobre la propia imagen, nos informa de las coordenadas asociadas al píxel en coordenadas imagen, así como a la clase a la que pertenece.

Figura 5.8: Resultado Segmentación

En el caso de haber optado por la opción *Asignación de colores aleatorios* obtendríamos la misma imagen pero con otros colores, Figura 5.9.

Figura 5.9: Resultado Segmentación aplicando colores aleatorios

Podemos aplicar un filtro que homogeniza la imagen y elimina el ruido subyacente. Para ello se irá recorriendo la imagen con una ventana de un determinado tamaño y se hará la media de los píxeles.

Para ello pulsamos el botón "*Aplicar filtro*". Nos saldrá una ventana emergente (Figura 5.10) que tiene por defecto el tamaño de ventana puesto a 3, dando la posibilidad de que el usuario modifique dicho valor a conveniencia:

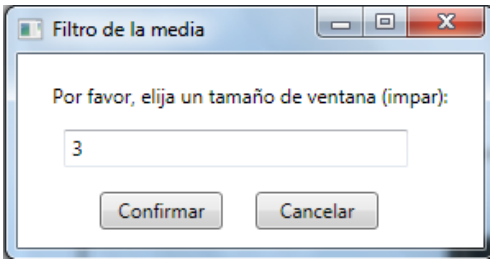


Figura 5.10: Ventana emergente para aplicar el filtro

La imagen resultante es la que aparece en la parte derecha de la Figura 5.11.

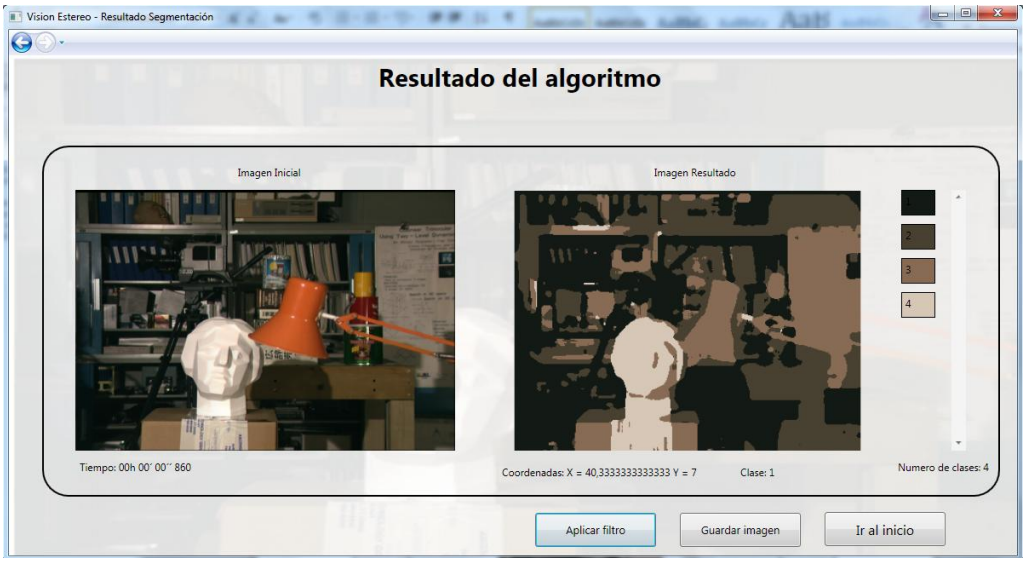


Figura 5.11: Resultado Segmentación aplicando el filtro

Podemos observar cómo en la nueva imagen resultado ya no existe tanto ruido, como consecuencia del filtrado realizado

Finalmente, si queremos, existe la opción de guardar la imagen resultado. Esto nos servirá como imagen base para los algoritmos que se describen a continuación.

5.4. Algoritmo de Correlación

Finalmente, pulsamos en el botón *Algoritmo de Correlación*, que aplica el algoritmo de su mismo nombre. Este algoritmo analiza la imagen resultado de segmentación de colores y la imagen resultado del Algoritmo de Lankton, obteniendo una nueva imagen mejorada sobre la que se eliminan la mayor parte de los píxeles originalmente erróneos generados por aquél.

Según se muestra en la pantalla que aparece en la Figura 5.12, existe la posibilidad de elegir dos opciones representadas por los respectivos botones:

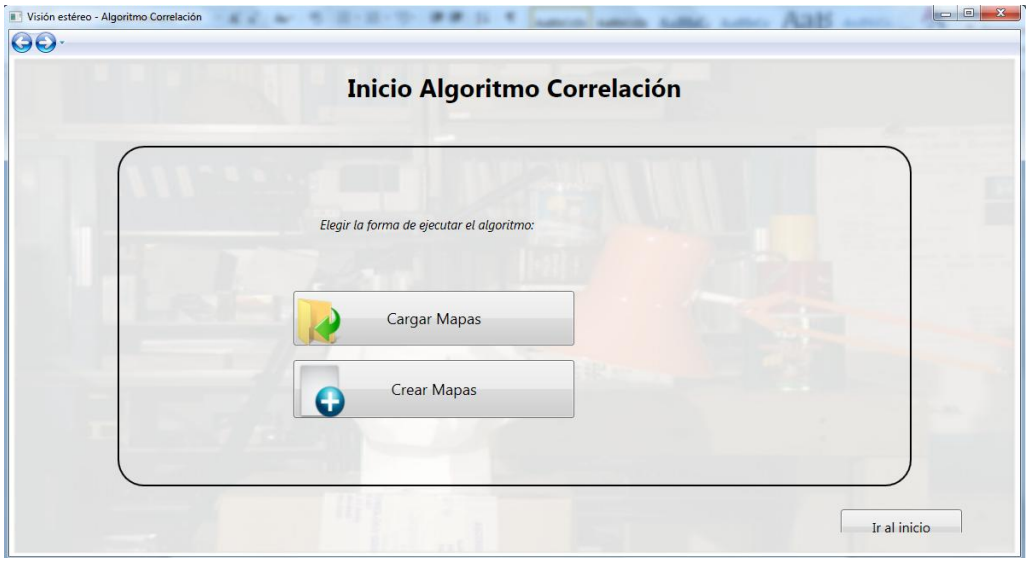


Figura 5.12: Pantalla para elegir la forma de ejecutar el Algoritmo de Correlación

1. **Cargar Mapas:** Si pulsamos sobre dicho botón nos dirigirá a una pantalla en la que podemos cargar los mapas de segmentación de colores y Lankton que previamente hemos guardado.

2. **Crear Mapas:** Este botón proporciona la opción de volver a generar los mapas de segmentación junto con la aplicación del algoritmo de Lankton.

Cargar Mapas:

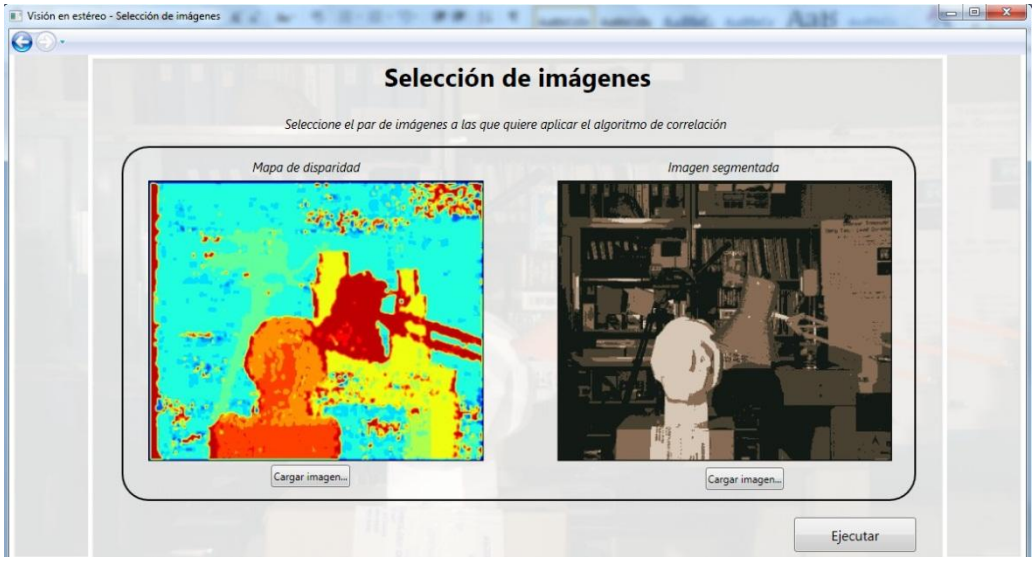


Figura 5.13: Pantalla de selección de imágenes de Segmentación y Algoritmo de Lankton

Cargamos las imágenes previamente guardadas. En la ventana izquierda se muestra la imagen resultado del algoritmo de Lankton, y en la imagen derecha la imagen resultado del algoritmo de segmentación de colores. Finalmente pulsamos "*Ejecutar*".

Aparece una ventana emergente (Figura 5.14) en la que podremos seleccionar el tamaño de la misma que irá recorriendo las dos imágenes para realizar el filtro, así como el tipo de filtro que queremos utilizar, moda, media o mediana.

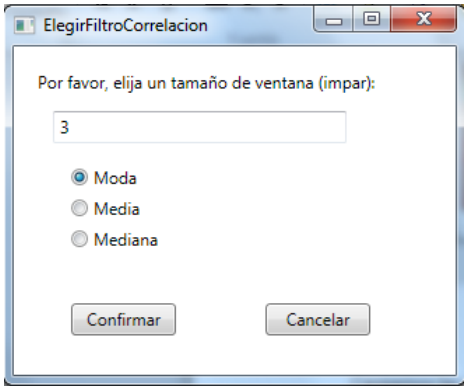


Figura 5.14: Ventana emergente para elegir el Filtro.

En los ejemplos mostrados a continuación elegimos tamaño de ventana 3.

Seguidamente se muestran los diferentes resultados que obtendremos al aplicar los distintos filtros. Podemos observar, tanto la imagen original, como la imagen resultado. De esta manera es posible observar las mejoras que se han obtenido con los distintos tipos de filtros.

Moda:

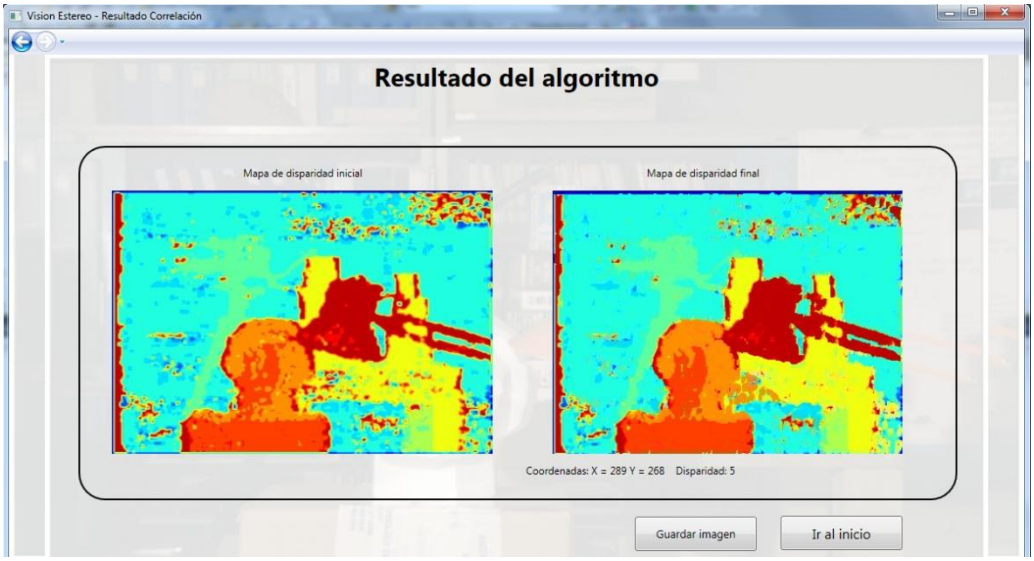


Figura 5.15: Pantalla resultado utilizando el filtro de la moda.

Media:

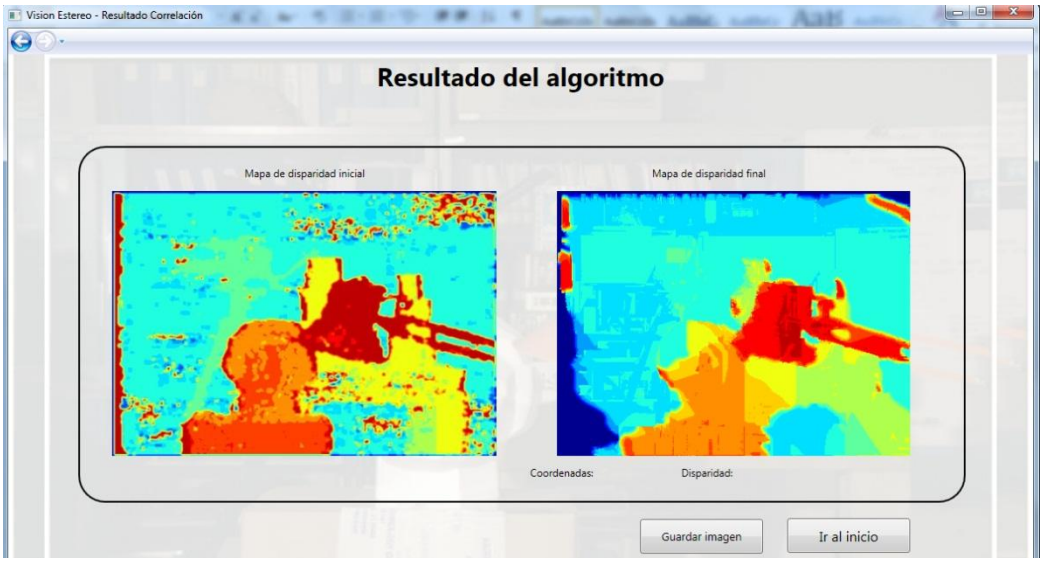


Figura 5.16: Pantalla resultado utilizando el filtro de la media

Mediana:

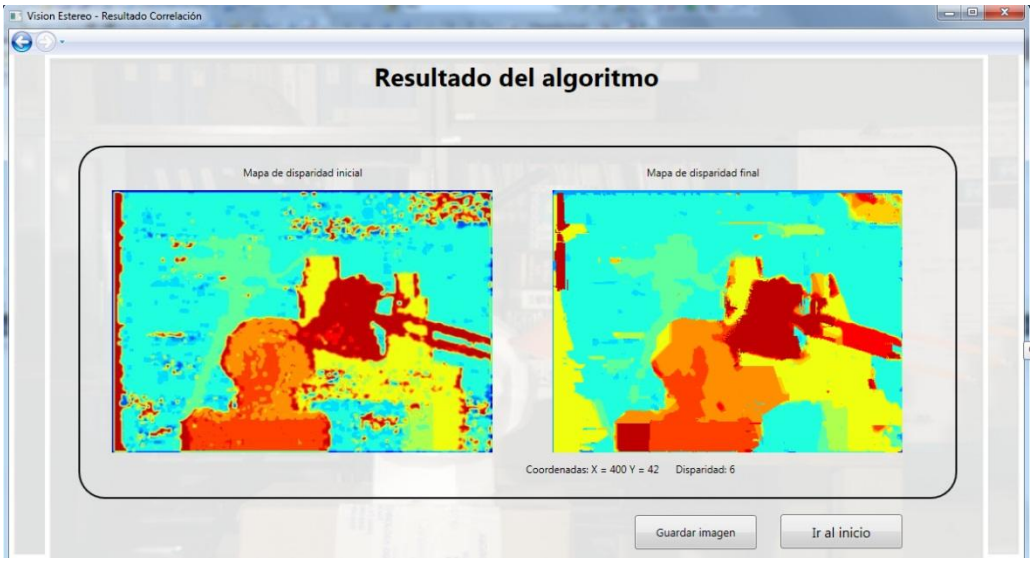


Figura 5.17: Pantalla resultado utilizando el filtro de la mediana

Crear Mapas:

Esta opción nos dirige a la pantalla que aparece en la Figura 5.18 con tres ventanas donde se irán cargando las imágenes según se vayan ejecutando los distintos algoritmos.

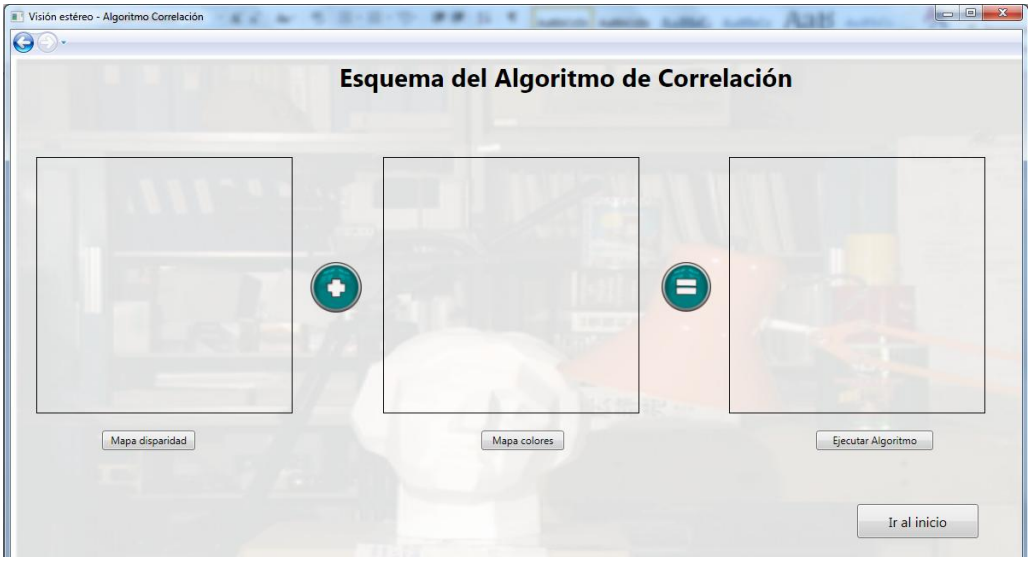


Figura 5.18: Esquema del algoritmo de correlación

Primeramente se pulsa la opción *Mapa disparidad*. Aquí volvemos a realizar exactamente los mismos pasos que con el botón *Algoritmo estéreo Lankton* con la diferencia de que en lugar de mostrar el resultado en una nueva pantalla, nos lo mostrará sobre la propia que se encuentra activa en este momento.

Seguidamente pulsamos en *Mapa colores* y al igual que en el caso anterior, volvemos a realizar los mismos pasos que los generados con el botón *Segmentación de Colores*, mostrando la imagen resultante sobre la misma pantalla activa.

Figura 5.19: Esquema del algoritmo de correlación con las imágenes de disparidad y segmentación cargadas

Finalmente pulsamos el botón *Ejecutar Algoritmo*.

De nuevo aparece una ventana emergente ofreciéndonos la opción para que elijamos el filtro y el tamaño de la ventana apropiado. En este sentido cabe la posibilidad de probar con los diferentes filtros.

Moda:

Figura 5.20: Esquema del algoritmo de correlación con la imagen resultado aplicando el filtro de la moda

Media:

Figura 5.21: Esquema del algoritmo de correlación con la imagen resultado aplicando el filtro de la moda

Mediana:

Figura 5.22: Esquema del algoritmo de correlación con la imagen resultado aplicando el filtro de la mediana

CAPÍTULO 6

PROCESO DE DESARROLLO

En esta sección se describen las distintas fases por las que ha pasado el proyecto y los retos y dificultades a los cuales hemos tenido que enfrentarnos en cada una de ellas. Durante cada una de las fases tuvimos reuniones periódicas con nuestro tutor, las cuales marcaron cada una de las fases y el propósito de las mismas:

- Primera fase:

La primera fase del proyecto duró hasta finales de octubre. Consistió, básicamente, en decidir cómo íbamos a desarrollar lo que se nos proponía como proyecto. Teníamos el proyecto que se desarrolló en Sistemas Informáticos durante el curso 2009/10, con el cual se consiguieron avances significativos, permitiendo desarrollar este proyecto a partir del suyo. Sin embargo, el proyecto completo estaba desarrollado en C#, un lenguaje el cual desconocíamos los miembros del grupo. La primera cuestión que teníamos que resolver era decidir si desarrollábamos desde cero una nueva aplicación en un lenguaje que ya conociéramos o continuar con el proyecto previo. Después de investigar a fondo el código de la aplicación, el propio lenguaje C# y las posibilidades que podían tener otros lenguajes, como Java, para desarrollar lo que queríamos, decidimos continuar con el proyecto del año pasado. La principal razón fue el hecho de que, nuestro proyecto precisaba de la implementación de un sistema de cálculo de disparidades como el que tenían ya desarrollada mediante el algoritmo de Lankton. Si queríamos comenzar una nueva aplicación, teníamos que desarrollar otro sistema al menos con las mismas prestaciones que el que ya estaba implementado. Nos pareció adecuado aplicar el principio de reusabilidad tal y como se concibe en el mundo empresarial.

- Segunda fase:

Una vez decidida la continuación con el proyecto anterior, comenzamos una segunda fase de estudio a fondo de la propia aplicación, tanto de su código como de las posibilidades que ofrecía. Esta fase terminó a mediados de Noviembre. Quedamos con los integrantes del grupo del año anterior para que nos ayudaran en la identificación de los problemas y nos resolvieran dudas de la implementación. Todo esto nos ayudó a comprender el manejo y el código de la aplicación, además de adquirir conocimientos acerca de C#. También nos vino bien su asesoramiento inicial de cara a prevenirnos sobre los principales problemas a los que nos íbamos a enfrentar.

En la memoria del proyecto del año pasado sugerían como líneas de futuro el estudio de la librería OpenCV para ampliar las funcionalidades de la aplicación. Sin embargo decidimos que para lo que queríamos hacer con el propio lenguaje C# nos bastaba, así que finalmente no se consideró.

- Tercera fase:

Una vez que ya conocíamos el proyecto en el que íbamos a trabajar y el lenguaje en que se iba a desarrollar, comenzamos con la propia implementación de nuestra parte de la aplicación. Lo primero que teníamos que hacer para implementar el algoritmo de correlación de Klaus era desarrollar un algoritmo que clasificara los colores de las imágenes por clases. Así que desarrollamos el algoritmo de segmentación de colores, ya explicado anteriormente, tomado de [PAJ07.1]. Esta fase nos llevó hasta finales de enero.

- Cuarta fase:

La cuarta fase se corresponde con la última fase de implementación y diseño de la aplicación. Desarrollamos el algoritmo de correlación, que hacía uso tanto de los resultados que devolvía el algoritmo de Lankton como el algoritmo de segmentación. Esta fase duró hasta mediados de abril.

- Quinta fase:

La última fase se dedicó a resolver fallos y detalles de la aplicación y a escribir la memoria del proyecto. También en esta fase buscamos otras imágenes estereoscópicas de cara a probar nuestra aplicación con imágenes diferentes y poder mostrar resultados diferentes.

CAPÍTULO 7

RESULTADOS Y CONCLUSIONES

Para concluir, en este capítulo mostraremos los resultados obtenidos mediante los algoritmos implementados en el sistema propuesto para los distintos tipos de imágenes con los que hemos realizado las pruebas.

7.1. Imágenes utilizadas

Las imágenes utilizadas para obtener los resultados de los algoritmos de nuestra versión del sistema de visión en estéreo fueron las siguientes:

- Imágenes del estudio de Middlebury [MID11]
- Imágenes de simulador
- Imágenes reales

7.1.1. Imágenes de Middlebury

Las imágenes estereoscópicas procedentes de la base de datos de de Middlebury son las más utilizadas. Son pues, imágenes ideales para conseguir resultados óptimos debido a que poseen las características idóneas para ello. La figura 7.1 y 7.2 muestran dos pares de imágenes obtenidas del banco de imágenes de Middlebury.

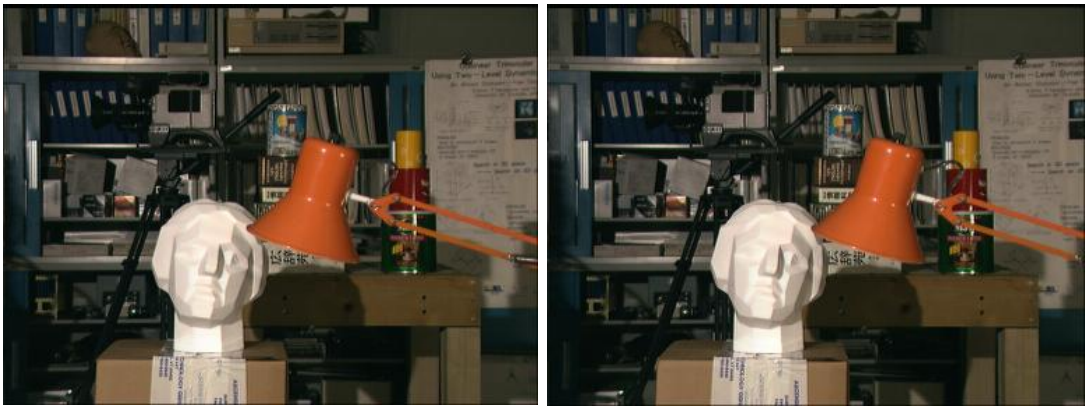


Figura 7.1: Imagen derecha e izquierda de Middlebury (busto)

Figura 7.2: Imágenes derecha e izquierda de Middlebury (conos)

7.1.2. Imágenes de simulador

Todos los resultados que se han obtenido con las imágenes de simulador se han basado en las imágenes mostradas en la Figura 7.3. Se han elegido estas imágenes para la realización de las pruebas pertinentes ya que poseen un fondo bien diferenciado del resto de la imagen. Sólo existe un objeto en la escena (el cubo) que está aislado, siendo fácilmente reconocible. Por tanto, esta imagen es una buena candidata para comprobar la efectividad de los algoritmos ya que no existen objetos que puedan generar errores en los resultados.

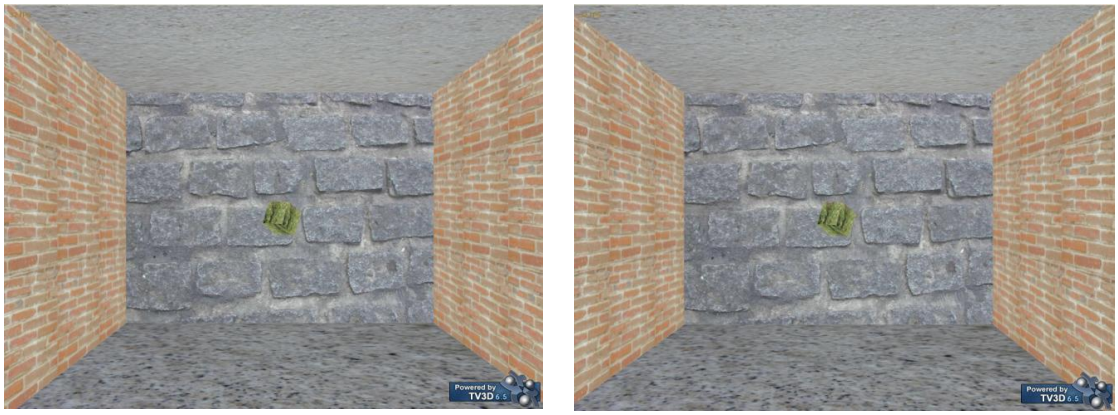


Figura 7.3: Imágenes de simulador

7.1.3. Imágenes reales

Las imágenes reales mostradas en la Figura 7.4 se han intentado obtener siguiendo el criterio de las imágenes del simulador, Figura 7.2. Esto es, tratando de mantener un fondo homogéneo y un único objeto en la escena. Estas imágenes fueron tomadas con un único dispositivo desplazado sobre la misma horizontal intentando simular la visión estereoscópica de las dos cámaras estereoscópicas de los sistemas de visión de esta naturaleza.

La dificultad principal que presentan estas imágenes es que los niveles de intensidad en las dos imágenes no son exactamente los mismos, por lo que a la hora de establecer las correspondencias aparece una problemática importante.

Figura 7.4: Imágenes reales

7.2. Resultados obtenidos

A continuación se presentan los resultados más significativos para cada uno de los algoritmos implementados con distintos tipos de imágenes. Los resultados varían en función de los parámetros introducidos y las opciones seleccionadas, tales como tamaño de ventana, tipo de filtro, etc.

Los resultados mostrados corresponden a imágenes segmentadas, en concreto, clasificadas por color siguiendo el Algoritmo de Cuantización Vectorial o bien, las imágenes correspondientes al mapa de disparidad, que se representa asignando colores cálidos (rojo, amarillo, naranja) a las regiones con valores de disparidad altos y que por tanto se encuentran más cerca del objetivo de las cámaras y en contraposición, asignando colores fríos (azul, cian...) a las regiones lejanas al objetivo que se caracterizan por tener valores bajos de disparidad.

7.2.1. Resultados del Algoritmo de Lankton

Los resultados del algoritmo de Lankton son mapas de disparidad representados con la asignación de colores fríos y cálidos en la imagen, tal y como se ha descrito previamente.

Con la versión mejorada del algoritmo de Lankton obtenemos resultados tanto mejores cuanto más diferenciados estén los objetos en la escena 3D.

En la Figura 7.5, para la imagen del busto de Middlebury, podemos apreciar con nitidez varios de los objetos y su cercanía respecto del sistema estereoscópico. En concreto, se puede distinguir varios objetos en el mapa de disparidad resultante. Podríamos concretar resaltando el hecho relevante de la figura del flexo en color naranja y el busto en color amarillo como los objetos más cercanos, y un fondo de colores azules para la parte de la escena que queda a mayor distancia del dispositivo de captura de las imágenes.

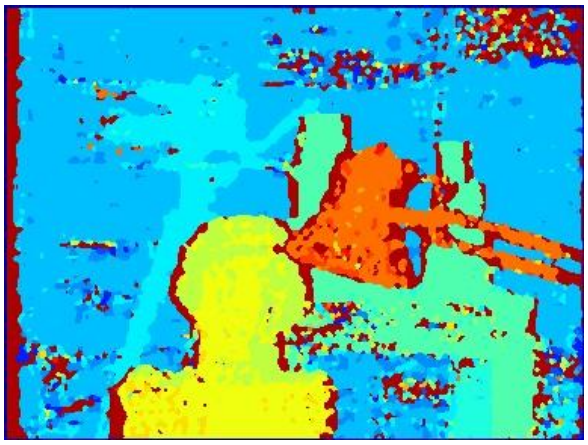


Figura 7.5. Resultados Lankton para imagen del busto

. Para la imagen de los conos vemos que la distancia de los píxeles entre la imagen izquierda y derecha sobre el eje horizontal es mayor. Para obtener el resultado de la Figura 7.6 hemos tenido que aumentar el valor del parámetro *disparidad máxima* a 60. Tampoco favorece al resultado la situación de los objetos de las imágenes originales que se encuentran a una distancia relativamente pequeña entre ellos.

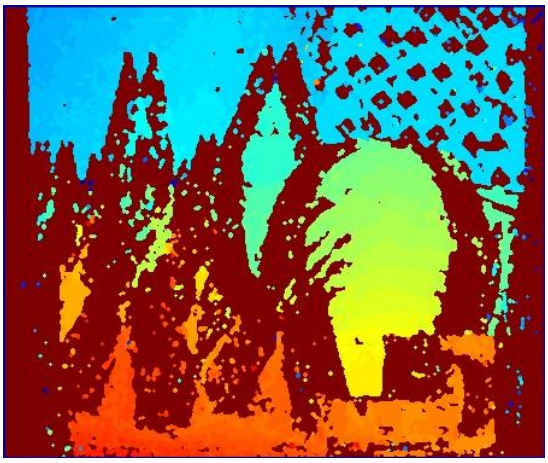


Figura 7.6. Resultados Lankton para la imagen de los conos.

Para las imágenes del simulador se obtienen muy buenos resultados. Se puede observar en la Figura 7.7 el cubo en el centro de la imagen con un color amarillo que indica una mayor proximidad a las cámaras y por tanto con valores mayores de disparidad, mientras que con un color azul verdoso aparece la pared del fondo de la imagen (valores menores de disparidad). Otra detalle observar en esta imagen es el degradado del rojo al azul verdoso que nos indica que cuanto más nos alejamos del sistema estereoscópico menor es la disparidad.

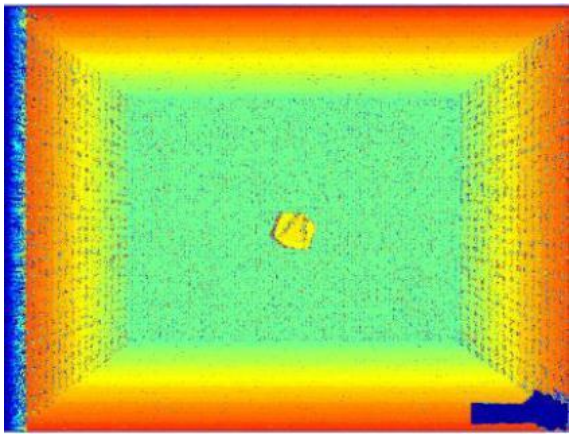


Figura 7.7. Resultados Lankton para imágenes de simulador

En el mapa de disparidad obtenido para las imágenes reales, Figura 7.8, a pesar de las impurezas, se pueden distinguir la vela y el bote como los objetos más cercanos, pues están coloreados de rojo y amarillo respectivamente. Detrás se distinguen el archivador y el espray con un color azul turquesa, por estar en un punto intermedio. All fondo, con un azul más intenso la pared y las sillas, como los objetos más alejados. Por lo comentado anteriormente y debido a la mala calidad de la imagen aparecen muchas impurezas las cuales consideramos como disparidades erróneas.

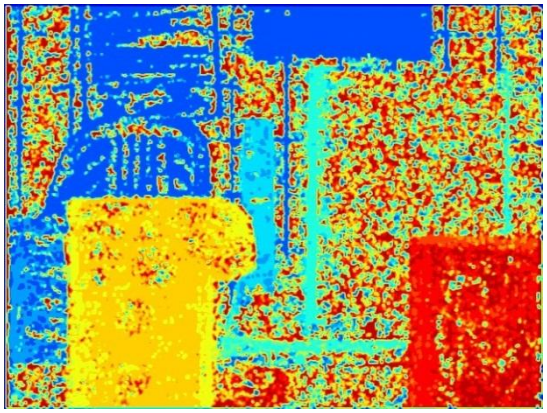


Figura 7.8. Resultados Lankton para imágenes reales

7.2.2. Resultados Segmentación de Color

Para segmentar las imágenes fuente por colores aplicamos el Algoritmo de Cuantización Vectorial implementado, obteniendo las correspondientes imágenes segmentadas como resultado. En ellas la característica principal estriba en el hecho de que se reduce considerablemente la gama de colores tras la clasificación o segmentación.

En función del valor umbral que fijemos previamente, obtenemos mayor o menor número de clases de colores. Cuanto más bajo sea el valor del umbral, mayor será el número de clases obtenidas.

Si aplicamos un umbral, por ejemplo con T=7, obtenemos 160 clases de colores. Como podemos observar en la Figura 7.9 la diferencia con la imagen original es prácticamente inapreciable.

Figura 7.9. Resultados Segmentación para un umbral T=7

Si aumentamos el valor del umbral a T=11, obtenemos un descenso considerable en el número de clases de colores de la imagen segmentada, tal y como puede observarse en la Figura 7.10, donde se reduce el número de clases a 22.

Figura 7.10. Resultados Segmentación para un umbral T=11

Para homogenizar la imagen, podemos aplicar un filtro de la moda sobre la imagen segmentada, obteniendo así los resultados de la Figura 7.11

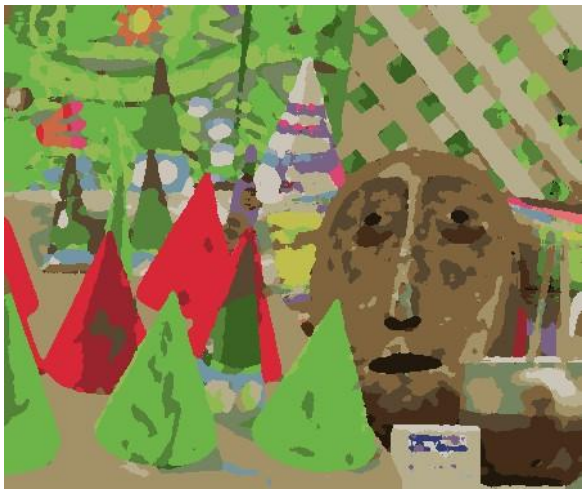


Figura 7.11. Resultados Segmentación con filtro aplicado.

En la imagen real anterior, para un umbral de T=10 tenemos el siguiente resultado (figura 7.12):

Figura 7.12. Resultado Segmentación para imágenes reales

Como la diferencia de colores es mínimamente apreciable, podemos resaltar la diferencia entre las clases asignando colores de forma aleatoria para conseguir así un contraste de intensidad mayor, como podemos apreciar en la Figura 7.13.

Figura 7.13. Resultados Segmentación para imágenes reales con colores aleatorios

7.2.3. Resultados del Algoritmo de Correlación

Los resultados que obtenemos al aplicar el Algoritmo de correlación son de nuevo imágenes que representan el mapa de disparidad. Esto es porque el objetivo fundamental de este algoritmo consiste en mejorar los resultados obtenidos con el Algoritmo de Lankton, utilizando como base la imagen segmentada en cada caso.

Para poder facilitar la tarea de comparación de las mejoras en los resultados, mostraremos una comparativa entre la imagen resultado del Algoritmo de Lankton y la imagen resultado del algoritmo de correlación implementado en el sistema que se describe en este trabajo.

Resultados para las imágenes de Middlebury

Para las imágenes procedentes de la base de datos Middlebury [MID11] (Figura 7.1), el Algoritmo de Correlación mejora sustancialmente el resultado obtenido por el Algoritmo de Lankton.

El resultado obtenido tras aplicar el filtro de la media resultó ser más satisfactorio de lo esperado. Podemos observar en la Figura 7.14 cómo consigue eliminar en gran parte el ruido de las regiones del flexo y el fondo de la imagen. Puede actuar de forma agresiva obteniendo grandes contrastes de disparidad dentro de una misma región, tal y como se observa en el objeto del busto.

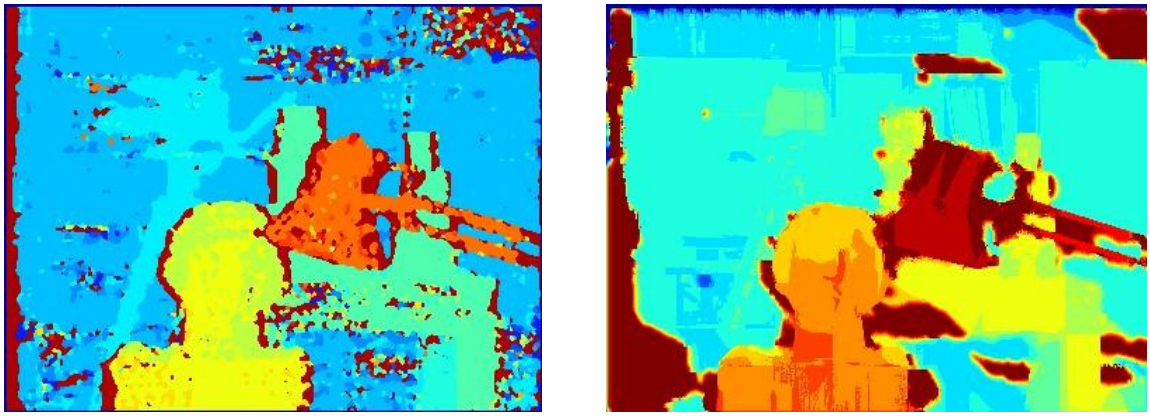


Figura 7.14. Comparativa resultados Algoritmo Correlación con el filtro de la media

Mediante la aplicación del filtro de la mediana podemos apreciar en la Figura 7.15 un hecho significativo, cual es que no respeta los bordes de los objetos, difuminándolos entre sí. Esto se puede observar claramente en el objeto del flexo.

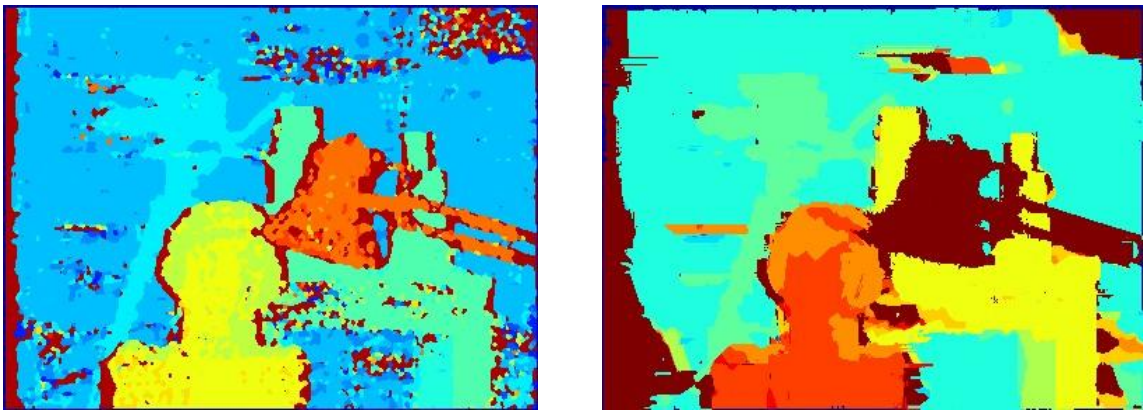


Figura 7.15: Comparativa resultados Algoritmo Correlación con el filtro de la mediana

En la Figura 7.16 observamos los resultados obtenidos al aplicar el filtro de la moda. Podemos apreciar que elimina la mayoría del ruido en las regiones, es decir, homogeniza estas regiones eliminando píxeles erróneos. El filtro de la moda respeta los bordes de las regiones y suaviza el resultado obtenido por el Algoritmo de Lankton.

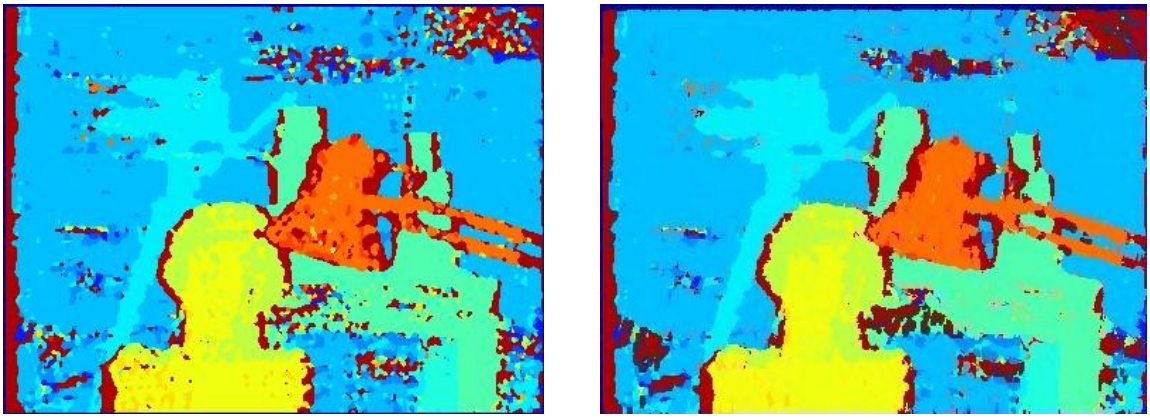


Figura 7.16. Comparativa resultados Algoritmo Correlación con el filtro de la moda

Para el mapa de disparidad obtenido con el Algoritmo de Lankton para la imagen de los conos, el resultado es difícil de mejorar debido a las limitaciones impuestas por el mapa de disparidad no satisfactorio obtenido en primer lugar por el Algoritmo de Lankton. En la Figura 7.17 podemos apreciar cómo aplicando el filtro de la moda en el Algoritmo de Correlación conseguimos homogenizar los bordes de los conos de la escena tridimensional 3D.

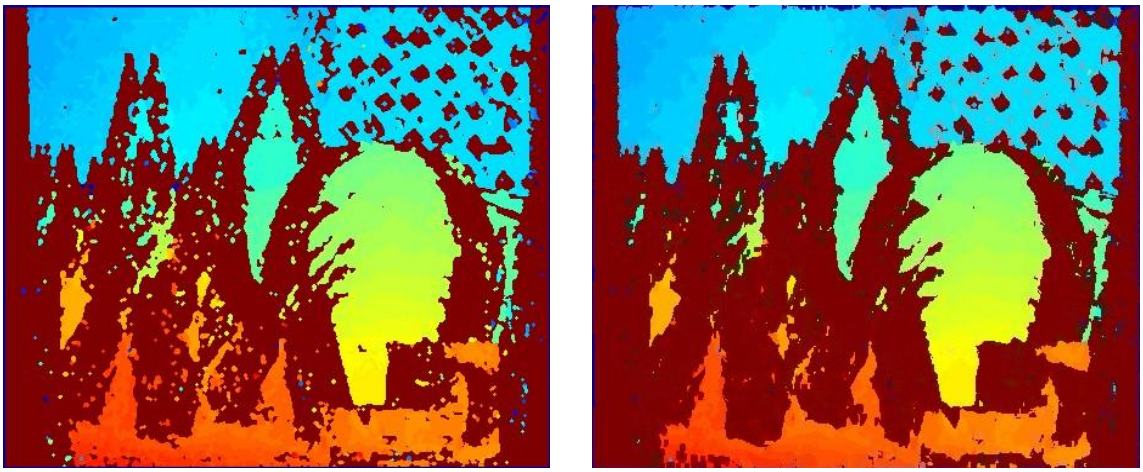


Figura 7.17: Comparativa resultados Algoritmo de Correlación para imagen de los conos con filtro de la moda

Como conclusión, observamos que el filtro de la moda aplicado en nuestro Algoritmo de Correlación resulta ser posiblemente la mejor forma de suavizar y corregir píxeles erróneos en estas imágenes, respetando la morfología de los objetos de la imagen original.

Resultados para imágenes reales

Para las imágenes reales contábamos con un mapa de disparidad muy heterogéneo obtenido con el Algoritmo de Lankton. Nuestro objetivo es filtrar, en la medida de lo posible, esas disparidades erróneas para obtener una imagen mejor.

Para el filtro de la moda con un tamaño de ventana de 3, conseguimos filtrar algunas disparidades erróneas, obteniéndose una ligera mejoría, un ligero suavizado de la imagen, pero apenas apreciable (figura 7.18).

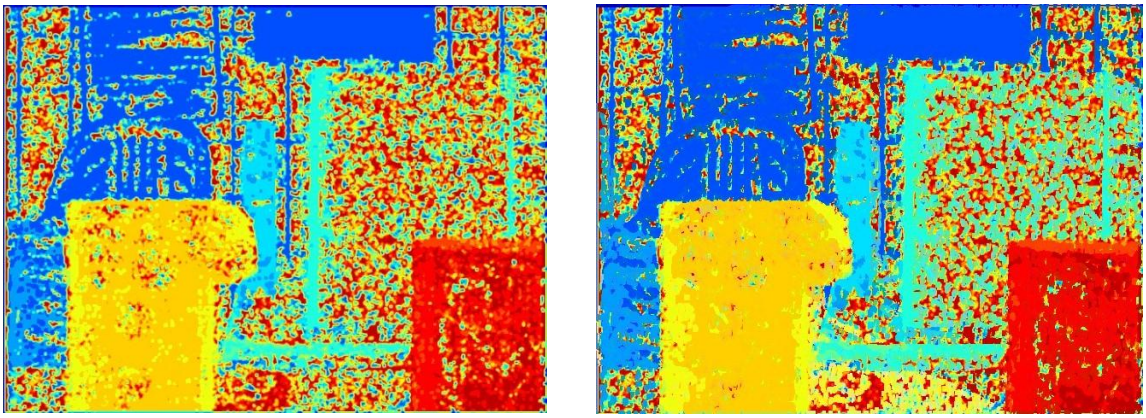


Figura 7.18: Comparativa resultados Algoritmo de Correlación para imágenes reales aplicando el filtro de la moda (ventana = 3).

Si aumentamos el tamaño de ventana para el filtro de la moda, observaremos que conseguimos filtrar muchas más impurezas, pero a costa de perder definición en los objetos (figura 7.19).

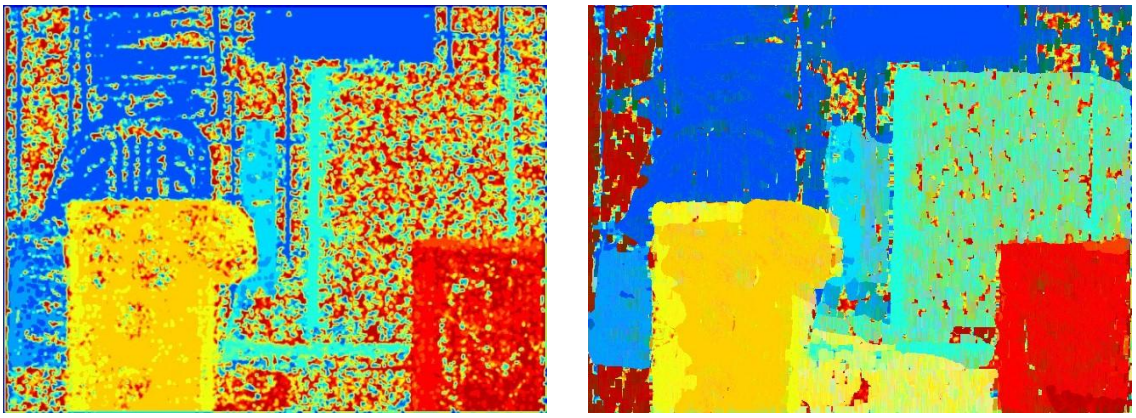


Figura 7.19: Comparativa resultados Algoritmo de Correlación para imágenes reales aplicando el filtro de la moda (ventana = 7).

Si usamos el filtro de la mediana obtendremos un buen suavizado de la imagen y una mayor nitidez en las formas de los objetos. Como contrapunto, ciertos colores de la imagen pueden verse mezclados y dar información errónea. En la figura 7.20, si nos fijamos en el archivador veremos que la esquina se ha pintado de amarillo, cuándo debería ser todo de color azul turquesa.

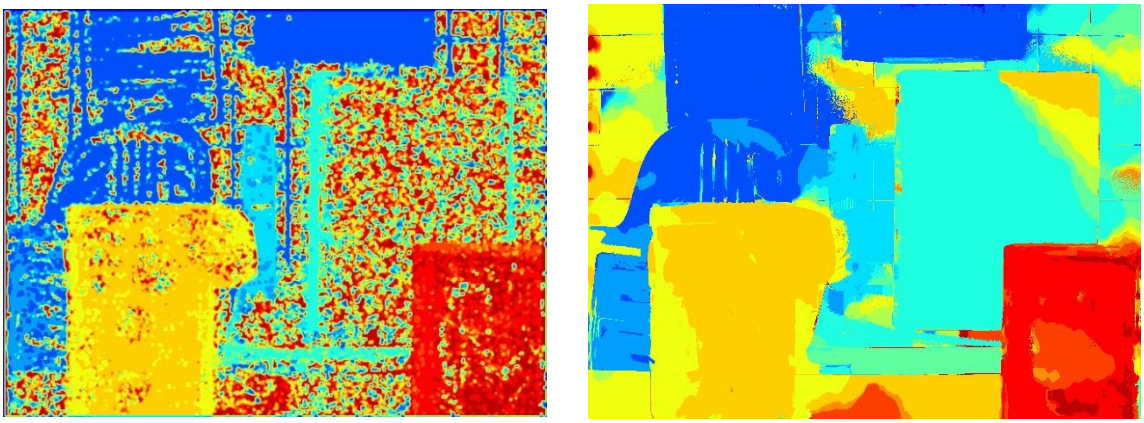


Figura 7.20. Comparativa resultados Algoritmo de Correlación para imágenes reales aplicando el filtro de la mediana.

Para concluir, podemos destacar los resultados obtenidos por el Algoritmo de Lankton para imágenes estereoscópicas no reales, es decir, no tomadas con el robot u otros dispositivos no calibrados.

Con estas imágenes se obtienen mejores resultados porque no cuentan con el brillo que se pueda obtener en una escena 3D real o la variación de intensidades de los píxeles entre la imagen izquierda y derecha consecuencia del calibrado de los dispositivos con los que son tomadas estas imágenes.

A la vista de los resultados obtenidos, podemos asegurar que el algoritmo de correlación mejora en gran medida los mapas de disparidad devueltos por el algoritmo de Lankton, contando siempre con la limitación del mapa a mejorar.

Las mejoras que obtenemos con los tres tipos de filtro que aplicamos en el Algoritmo de Correlación suavizan las regiones de la imagen que representa el mapa de disparidad, eliminando los píxeles erróneos. Observando los resultados obtenidos para los distintos tipos de imágenes utilizadas en los casos de prueba, podemos llegar a la conclusión que en función de cada imagen, será más adecuado usar un filtro u otro. Para las imágenes de Middlebury e ideales se obtiene una gran mejoría mediante el filtro de la moda y excesivos filtrados con el de la media y la mediana. Si bien, para imágenes reales con muchas más impurezas, obtenemos escasos resultados con la moda, pero buenos resultados con los filtros de la media y la mediana.

APÉNDICE A

INTERFAZ DE LA PROGRAMACIÓN DE LA APLICACIÓN

Espacio de nombres

Dentro de este espacio de nombres se sitúan los métodos principales con los algoritmos que incorpora nuestra aplicación:

- **AlgoritmoLankton2** (Bitmap imagenI, Bitmap imagenD, Configuracion config)
- **Anaglifo** (Bitmap imagenI, Bitmap imagenD)
- **EjecutarAlgCuantizacionVectorial**(Bitmap imagenI, Bitmap imagenD, Configuracion config)
- **EjecutarAlgCorrelacion**(Bitmap mapaDisp, Bitmap imagenSegm, int[,] mapaDisparidades, int[,] mapaClases)

A continuación se describen los métodos:

AlgoritmoLankton2 (Bitmap imagenI, Bitmap imagenD, Configuracion config)

Breve descripción

Método principal para ejecutar la segunda versión del algoritmo de correspondencia de Lankton.

Valor de retorno

Un Bitmap con el resultado del algoritmo aplicado al par de imágenes.

Parámetros

imagenI Imagen izquierda
imagenD Imagen derecha
config Parámetros de configuración del algoritmo

Bitmap Anaglifo (Bitmap imagenI, Bitmap imagenD)

Breve descripción

Método principal para mostrar un anaglifo a partir de dos imágenes.

Valor de retorno

Un Bitmap con el resultado del algoritmo aplicado al par de imágenes.

Parámetros

imagenI Imagen izquierda

imagenD Imagen derecha

Bitmap EjecutarAlgCuantizacionVectorial(Bitmap imagen, int umbral, bool coloresAleatorios)

Breve descripción

Método principal donde se ejecuta el algoritmo de Cuantización vectorial para realizar la segmentación de colores.

Valor de retorno

Un Bitmap con el resultado de aplicar el algoritmo sobre la imagen

Parámetros

Bitmap Imagen

Int umbral

Bool coloresAleatorios

Bitmap EjecutarAlgCorrelacion(Bitmap mapaDisp, Bitmap imagenSegm, int[,] mapaDisparidades, int[,] mapaClases)

Breve descripción

Método principal donde se ejecuta el algoritmo de correlación

Valor de retorno

Un Bitmap con el resultado de aplicar el algoritmo sobre la imagen izquierda (resultado Lankton) y la imagen derecha (resultado segmentación de colores)

Parámetros

Bitmap mapa Disparidad

Bitmap mapa Segmentación

Int[,] matriz Disparidad

Int[,] matriz Segmentación

BIBLIOGRAFÍA

- [KLA06] A. Klaus, M. Sormann, and K. Karner. "Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure", 2006.

- [LAN07] Shawn Lankton. "3D Vision with Stereo Disparity", 2007.

- [PAJ07.1] G. Pajares and J.M. De la Cruz. "Visión por computador. Imágenes digitales y aplicaciones". Ra-Ma, 2007.

- [PAJ07.2] G. Pajares and J.M. De la Cruz. "Ejercicios resueltos de visión por computador". Ra-Ma, 2007.

- [MID11] "Middlebury Stereo Vision Page" http://vision.middlebury.edu/stereo/ (accedido 5 Junio 2011)

- [ARL06] J.Arlow, I.Neustadt. "UML 2". ANAYA, 2006.